

Fundamental Data Science for Data Scientist

Veerasak Kritsanapraphan - Software Park Thailand

Data Science Series

- Data Science for Business
- Practical Data Science with Rapid Miner
- Basic R Programming
- Fundamental Data Science for Data Scientist

Agenda

Day 1

- Introduction to Data Science
 - What is Data Science?
 - Brief History of Data Analysis
 - What make data science different?
 - Data Science as Strategy and a Management Discipline
 - Reasoning, Decision Making and Relevance to Data Science
 - Discussion
- Data Scientist
 - Role of the Data Scientist
 - How to setup Data Science team
 - What Does It Take To Create A Data Science Capability?
 - Discussion

Agenda - Day 1

- **Data Science Life Cycles**
 - A Strategy to Approach Any Data Analytics Problems
 - Identify problems
 - Identify Data Sources
 - Identify Additional Data Sources
 - Statistic Analysis
 - Development and Implementation
 - Communicating Result
 - Maintenance
 - Discussion

Agenda - Day 2

- Component Parts of Data Science and Engineering a Data Science Solution
 - Overview of Data Engineering and Big Data Technologies
 - The Data Scientist's Toolbox – Languages, Platforms, Tools and Industry Solutions
- Basic Data Science Methods and Machine Learning Fundamentals
 - Regression
 - Decision Tree
 - K-Nearest Neighbor
 - Naive Bayesian Classifier
 - Association Rules
 - Other advanced topics
 - Discussion

Agenda

Day 2

6. Introduction to R/Python

- R/Python for Data Input and Output
- Data Frame
- Basic Operation/Function
- Discussion

Day 3

7. Getting and Cleaning Tidy using R

- Concept of Tidy Data
- Data preparation for Data Science
- Import and Export Data in Excel, CSV, XML, JSON
- Import and Export Data from Hadoop
- Discussion



8. Exploratory Data Analysis

- Statistic Summary
- Visualization
- Clustering data
- Discussion

9. Practical Machine Learning

- Predictive Analytics
- Classification Problems
- Association Rules Problems
- Big Data Problems
- Discussion

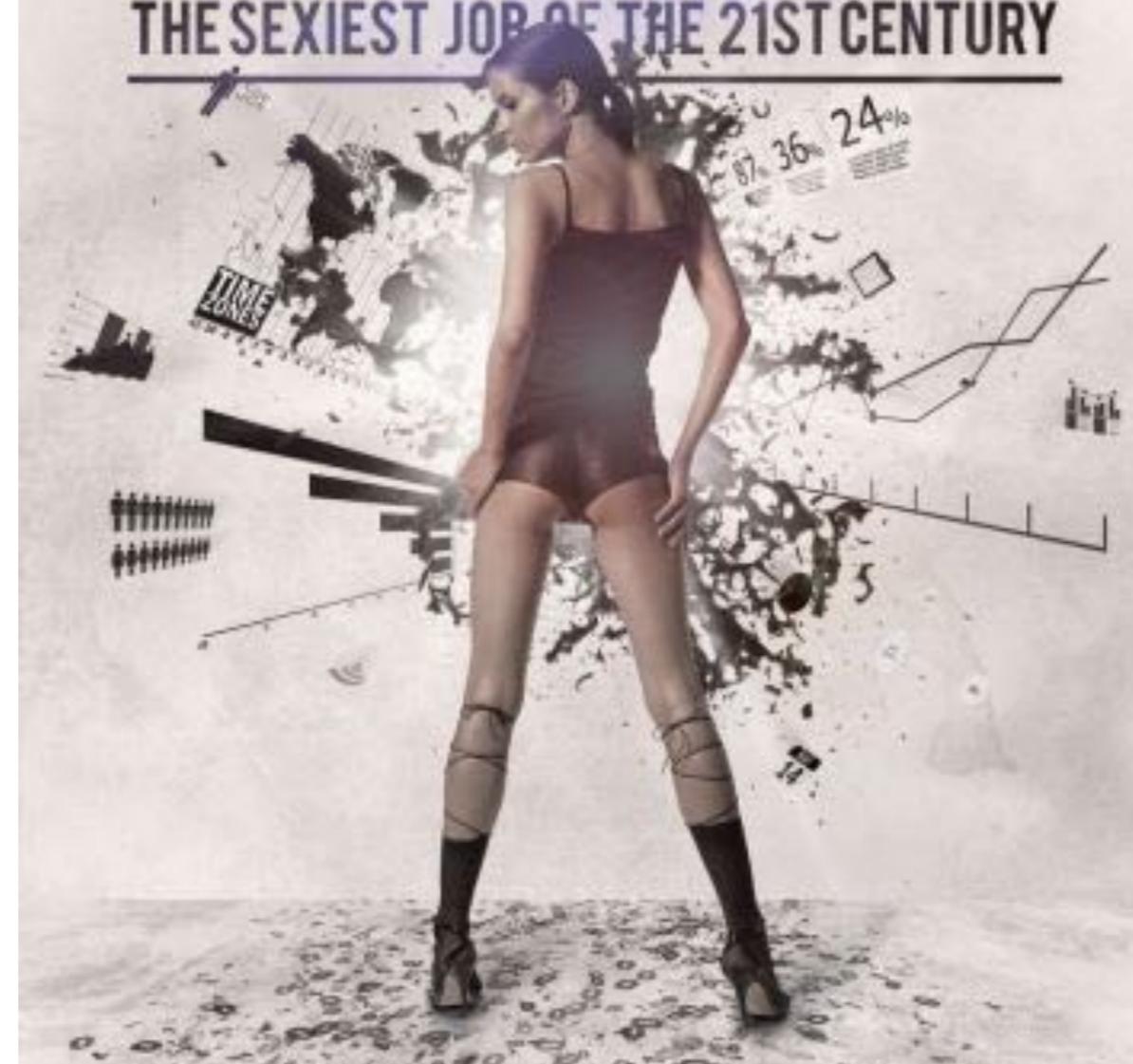
10. Wrap-up

About myself?

- Graduated from San Francisco State University in Master of Science in Computer Information System, 1997
- PhD. Candidate at Chulalongkorn University, research focus on Data Science, Big Data, Mobile Computing and Internet of Thing (IOT)
- Assistant Chief Information Officer at Bangkok Hospital Group
- Instructor for Software Park in Data Science, Requirement Discovery and Practical Enterprise Integration
- Instructor for Chulalongkorn University in R and Data Mining

DATA SCIENTIST

THE SEXIEST JOB OF THE 21ST CENTURY



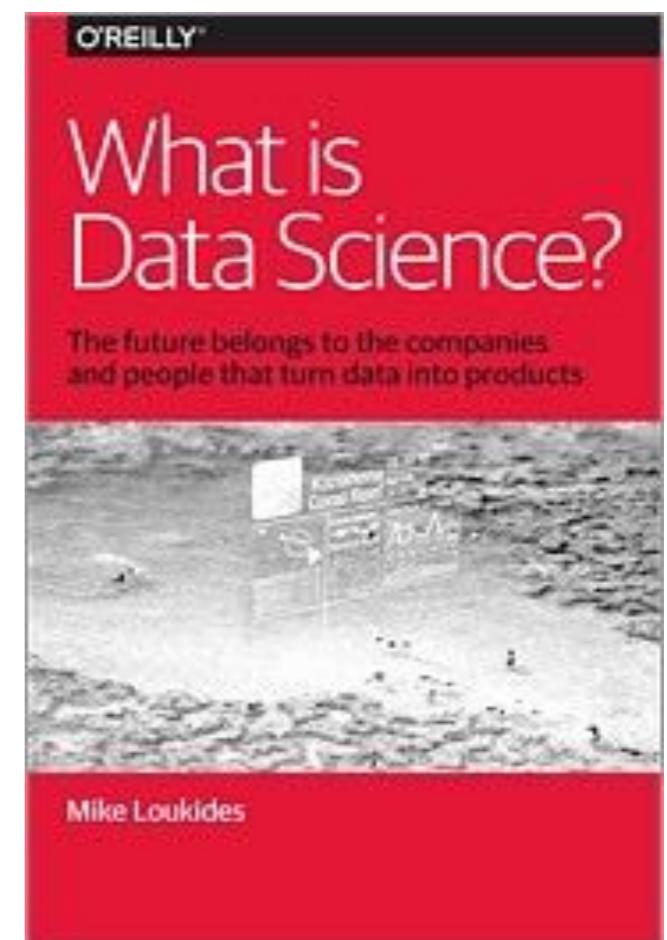
Introduce yourself?

- What is your name/position/role in your organization?
- Tell you a bit about yourself?
- What is your passion?
- What is your expectation for this class?



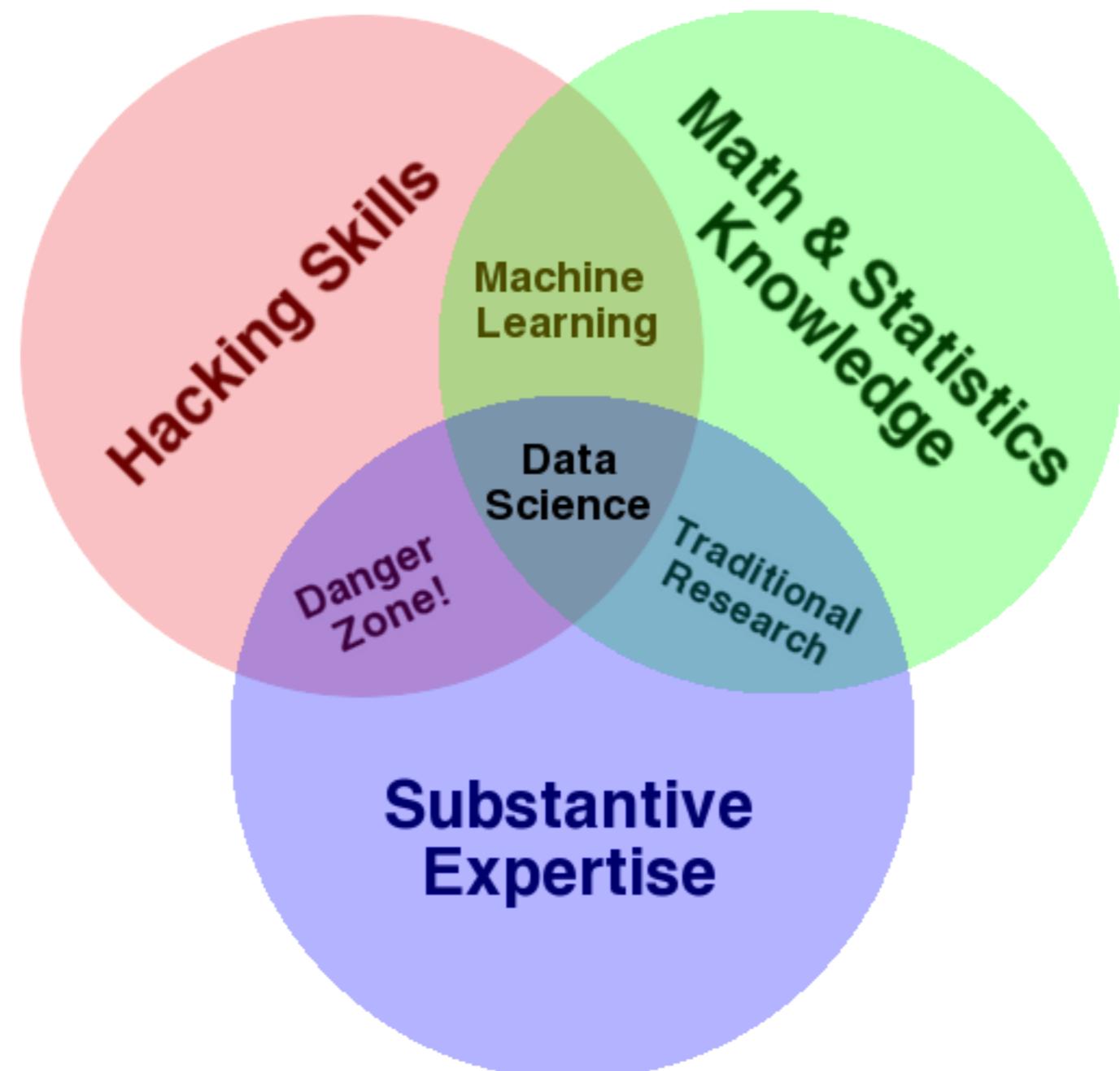
What is Data Science?

- Data Science aims to derive **knowledge** from **big data, efficiently and intelligently**
- Data Science encompasses the **set of activities, tools, and methods** that enable **data-driven activities** in science, business, medicine, and government



<http://www.oreilly.com/data/free/what-is-data-science.csp>

Data Science - Drew Convey's Definition



Data Science vs. Databases

Element	Databases	Data Science
WithValue	Precious	Cheap
DataVolume	Modest	Massive
Examples	Bank records, Personnel records, Census, Medical records	Online clicks, GPS logs, Tweets, tree 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	Structured Query Language (SQL)	NoSQL : Riak , Memcached , Apache Hbase , Apache River , MongoDB , Apache Cassandra , Apache CouchDB , ...

*CAP = Consistency, Availability, Partition Tolerance"

+ACID = Atomicity, Consistency, Isolation and Durability"

Data Science vs. Databases

Databases

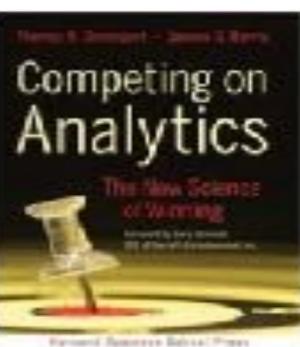
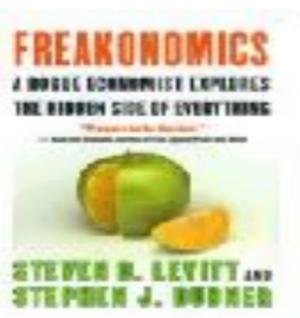
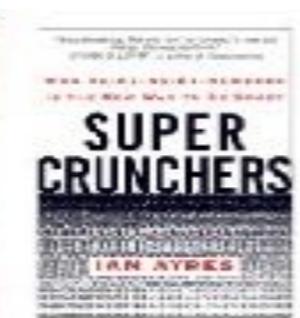
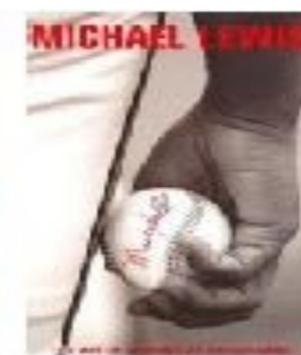
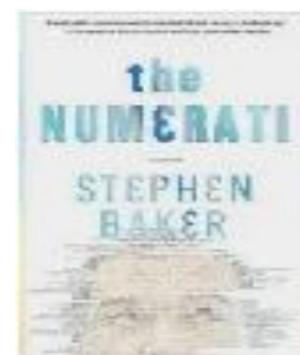
Querying the past

Data Science

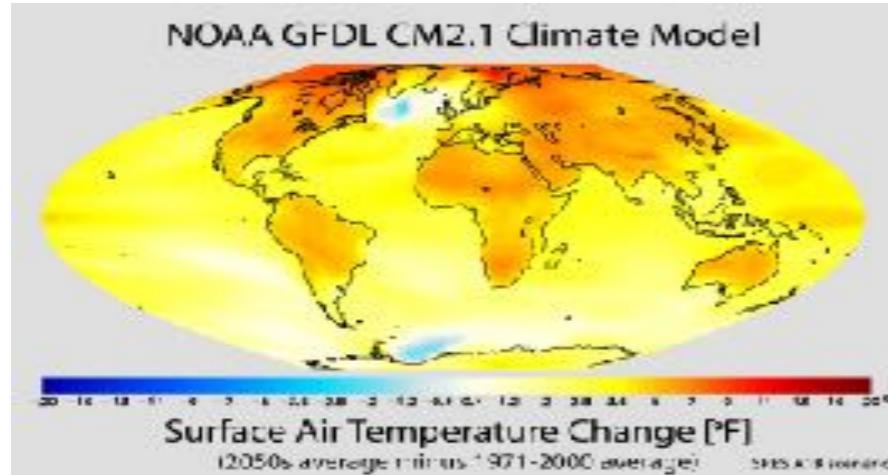
Querying the future

Related – Business Analytics

- » Goal: obtain “actionable insight” in complex environments”
- » Challenge: vast amounts of disparate, unstructured data and limited time”



Data Science vs. Scientific Computing - General purpose ML classifier



Scientific Modeling	Data-Driven Approach
Physics-based models	General inference engine replaces model
Problem-Structured	Structure not related to problem
Mostly deterministic, precise	Statistical models handle true randomness, and unmodeled complexity
Run on Supercomputer or High-end Computing Cluster	Run on cheaper computer Clusters (EC2)

Data Science vs. Traditional Machine Learning

Traditional 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!

History of Data Science

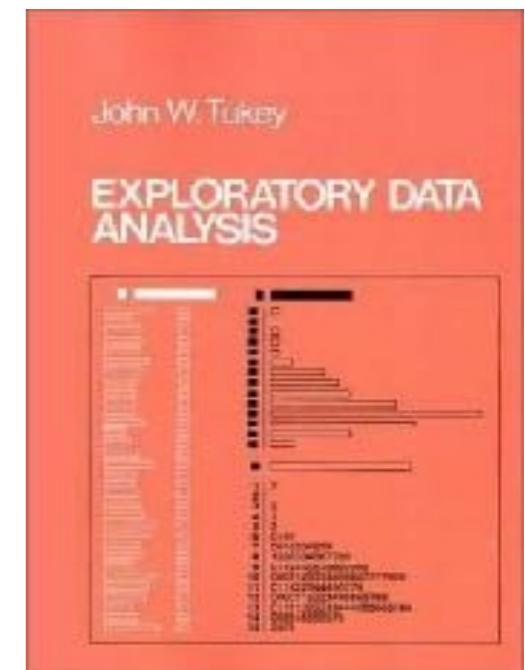
- R.A. Fisher
 - » 1935: “The Design of Experiments”
“correlation does not imply causation”
- W. E. Demming
 - » 1939: “Quality Control”



Images: <http://culturacientifica.wikispaces.com/CONTRIBUCIONES+DE+SIR+RONALD+FISHER+A+LA+ESTADISTICA+GENETICA>
http://es.wikipedia.org/wiki/William_Edwards_Deming

Brief Data Analysis History

- Peter Luhn
 - » 1958: "A Business Intelligence System"
- John W. Tukey
 - » 1977: "Exploratory Data Analysis"
- Howard Dresner
 - » 1989: "Business Intelligence"



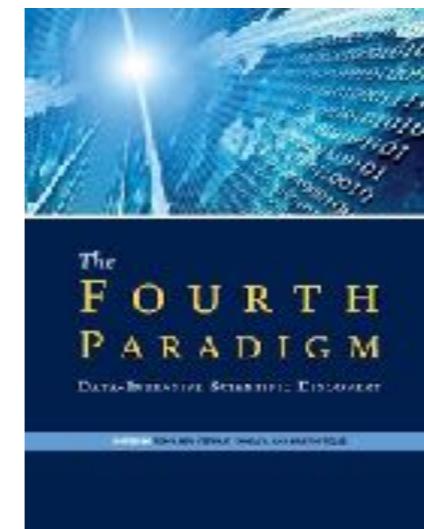
Images: <http://www.businessintelligence.info/definiciones/business-intelligence-system-1958.html> <http://www.betterworldbooks.com/exploratory-data-analysis-id-0201076160.aspx>
<https://www.flickr.com/photos/42266634@N02/4621418442>

Brief Data Analysis History

- Tom Mitchell
 - » 1997: "Machine Learning book"
- Google
 - » 1996: "Prototype Search Engine"
- Data-Driven Science eBook
 - » 2007: "[The Fourth Paradigm](#)"



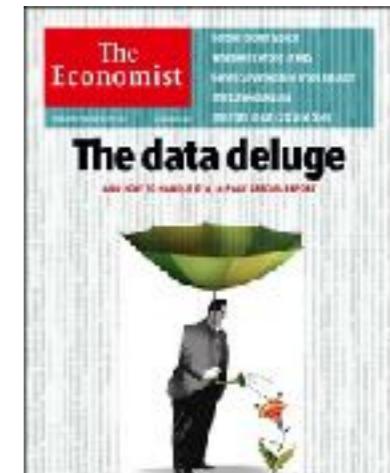
Google



Images: <http://www.amazon.com/Machine-Learning-Tom-M-Mitchell/dp/0070428077> <http://www.google.com/about/company/history/> <http://research.microsoft.com/en-us/collaboration/fourthparadigm/>

Brief Data Analysis History

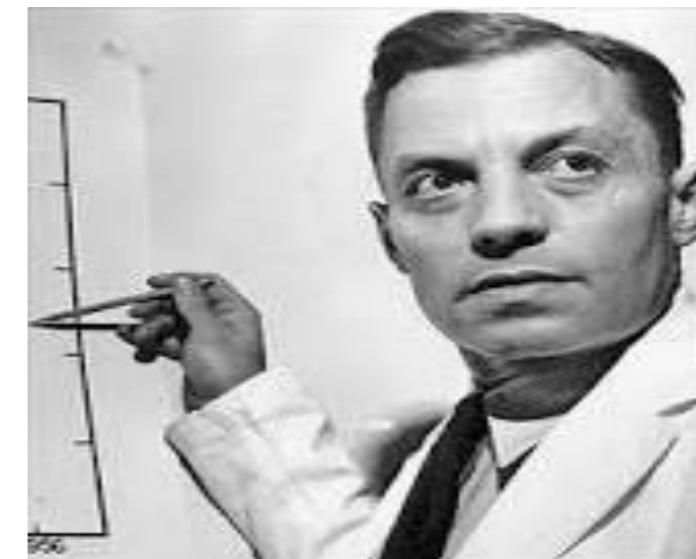
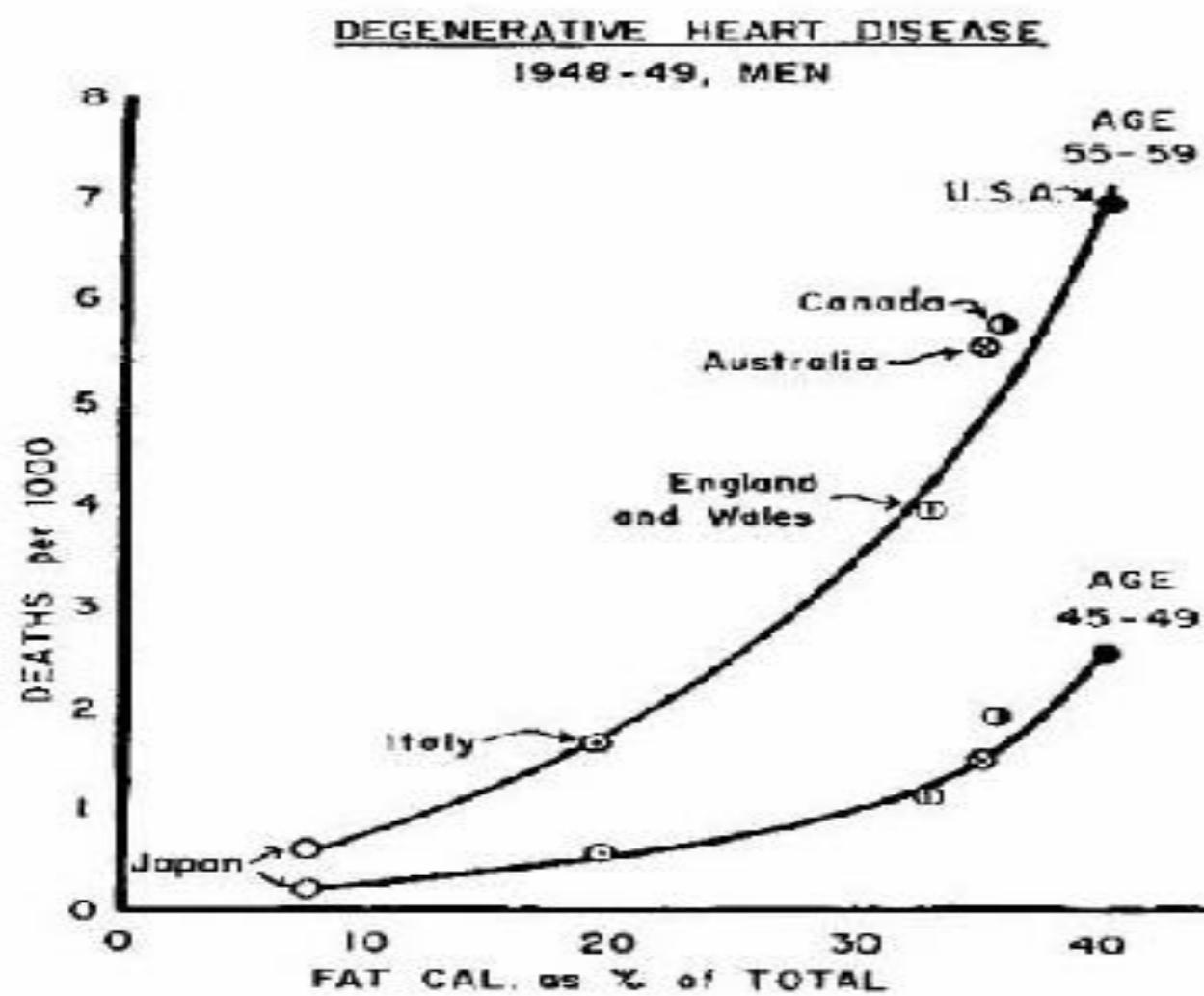
- Peter Norvig
 - » 2009: ‘The Unreasonable Effectiveness of Data’
- Exponential growth in data volume
 - » 2010: ‘The Data Deluge’



Images: http://en.wikipedia.org/wiki/Peter_Norvig
<http://www.economist.com/node/15579717>

Data Science?

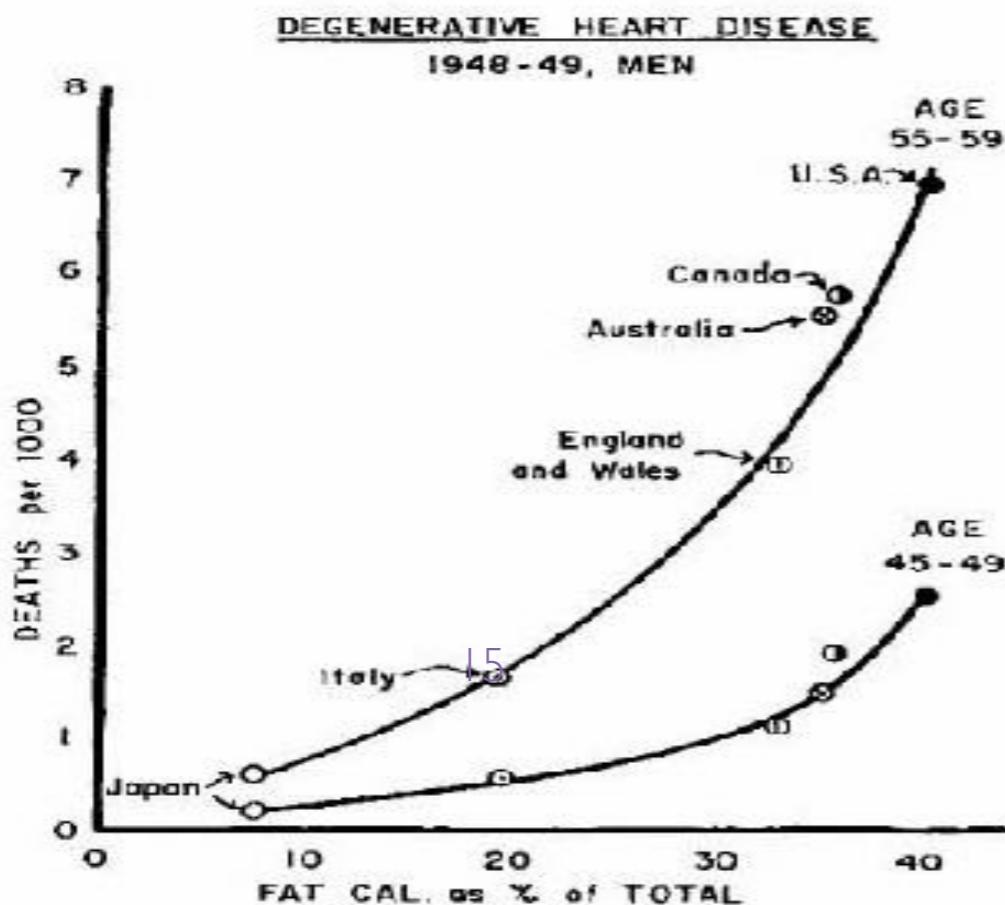
- Seven Countries Study (Ancel Keys)
 - » Started in 1958, followed 13,000 subjects total for 5-40 years



http://en.wikipedia.org/wiki/Seven_Countries_Study

Data Science?

- Seven Countries Study (Ancel Keys)
 - » Started in 1958, followed 13,000 subjects total for 5-40 years



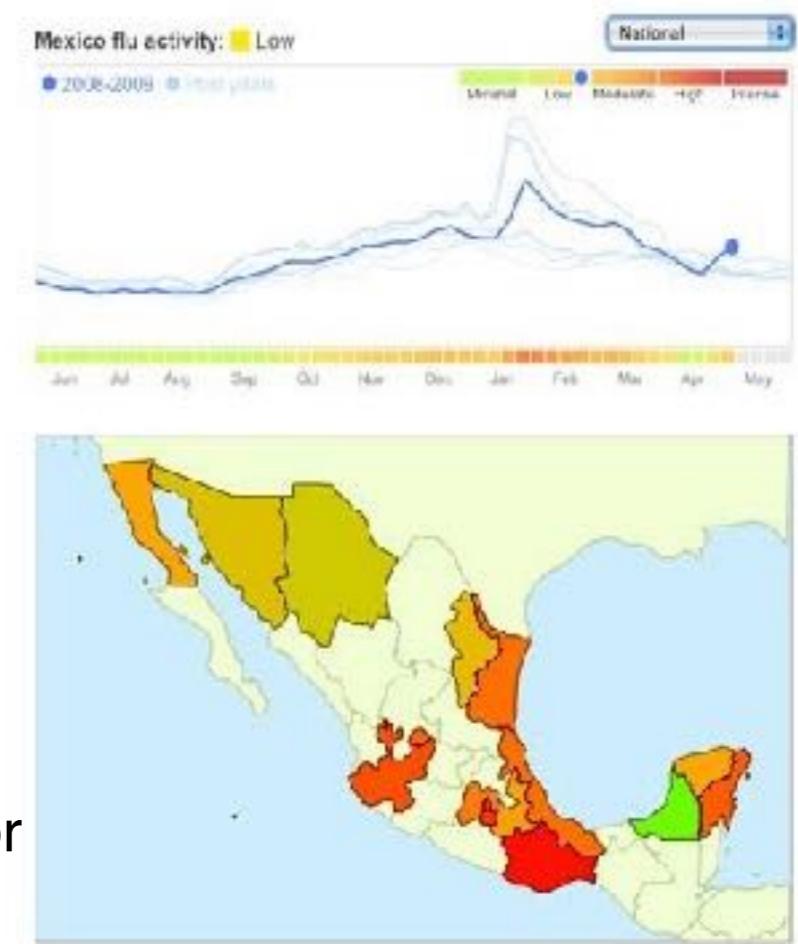
Significant controversy

- Only studied subset of 21 countries with data
- Failed to consider other factors (e.g., per capita annual sugar consumption in pounds)

http://en.wikipedia.org/wiki/Seven_Countries_Study

Data Science?

- Nowcasting vs Forecasting
- Example – Google Flu Trends:
 - » February 2010 detected outbreak two weeks ahead of CDC data
 - » Initially 97% accurate but overestimated during 2011-13 including one interval in 2012-13 period where GFT was off by 2x
 - » New models are estimating which cities are most at risk for spread of the Ebola virus



<https://www.google.org/flutrends/>

Data Science?

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



**USA 2012
Presidential
Election**

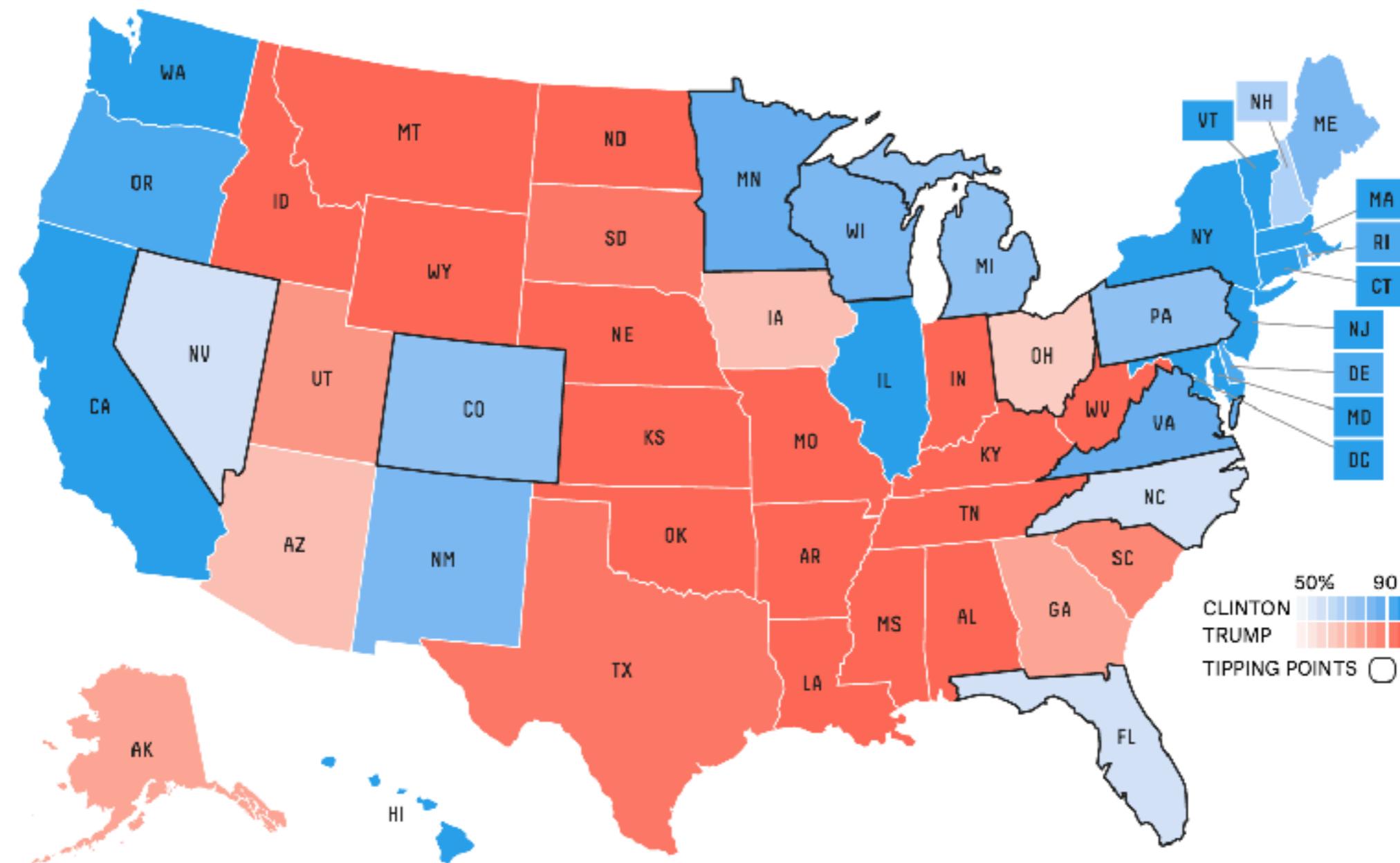
*the signal and the noise
and the noise and the noise
noise and the noise
why most noise and
predictions fail to predict
but some don't predict
and the noise and the noise
the noise and the noise
nate silver noise
noise and the noise*

<http://www.theguardian.com/world/2012/nov/07/nate-silver-election-forecasts-right>

Who will win the presidency?



Chance of winning



Electoral votes

■ Hillary Clinton	302.2
■ Donald Trump	235.0
■ Evan McMullin	0.8
■ Gary Johnson	0.0

Popular vote

■ Hillary Clinton	48.5%
■ Donald Trump	44.9%
■ Gary Johnson	5.0%
■ Other	1.6%



NOV 6, 2016 AT 1:10 PM

Election Update: Don't Ignore The Polls — Clinton Leads, But It's A Close Race

By [Nate Silver](#)

Filed under [2016 Election](#)





306 trump ✓

46.2% votes | 62,955,363



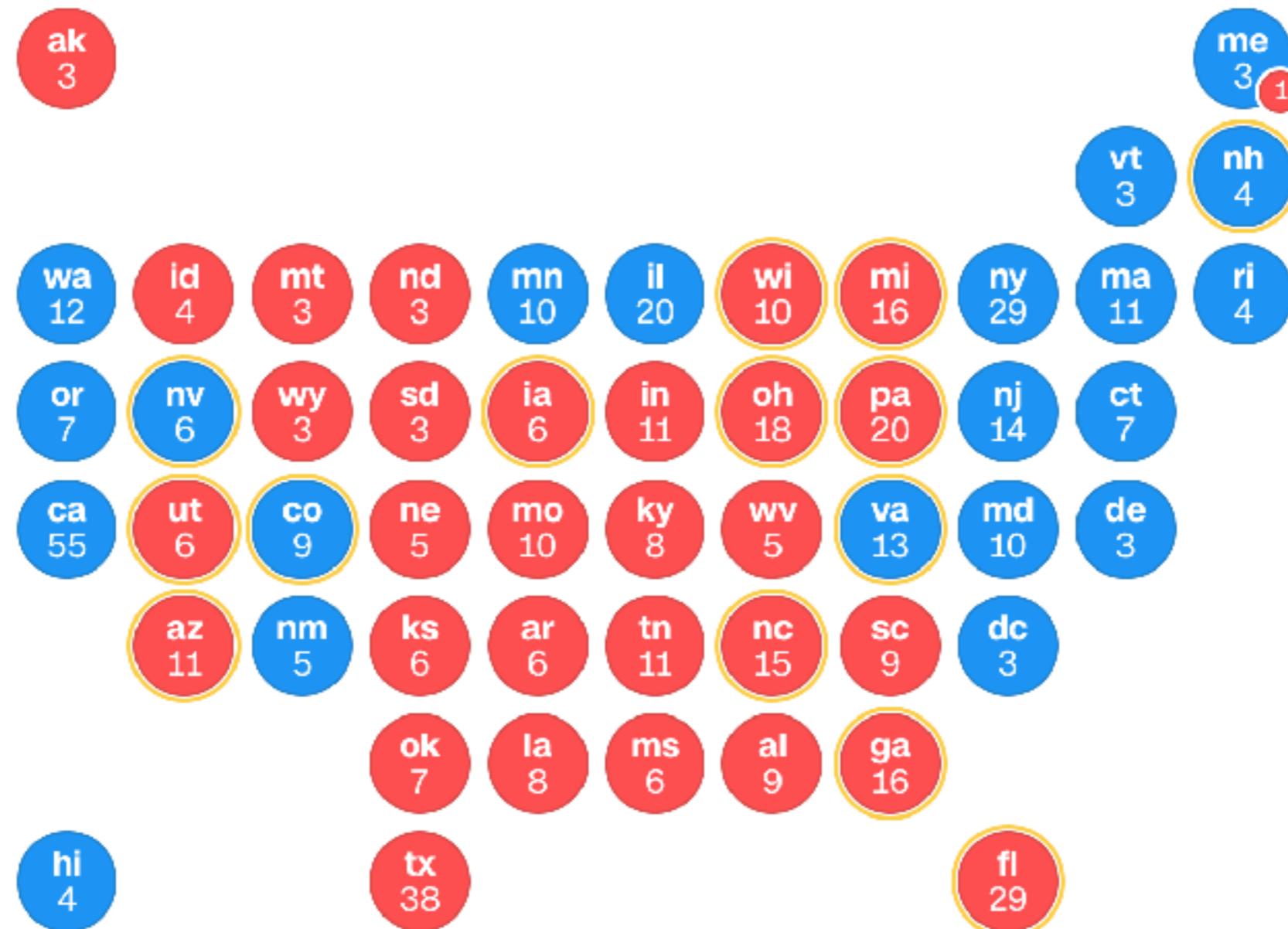
clinton 232

65,788,583 | 48.3% votes

270 electoral votes to win

national map

popular vote



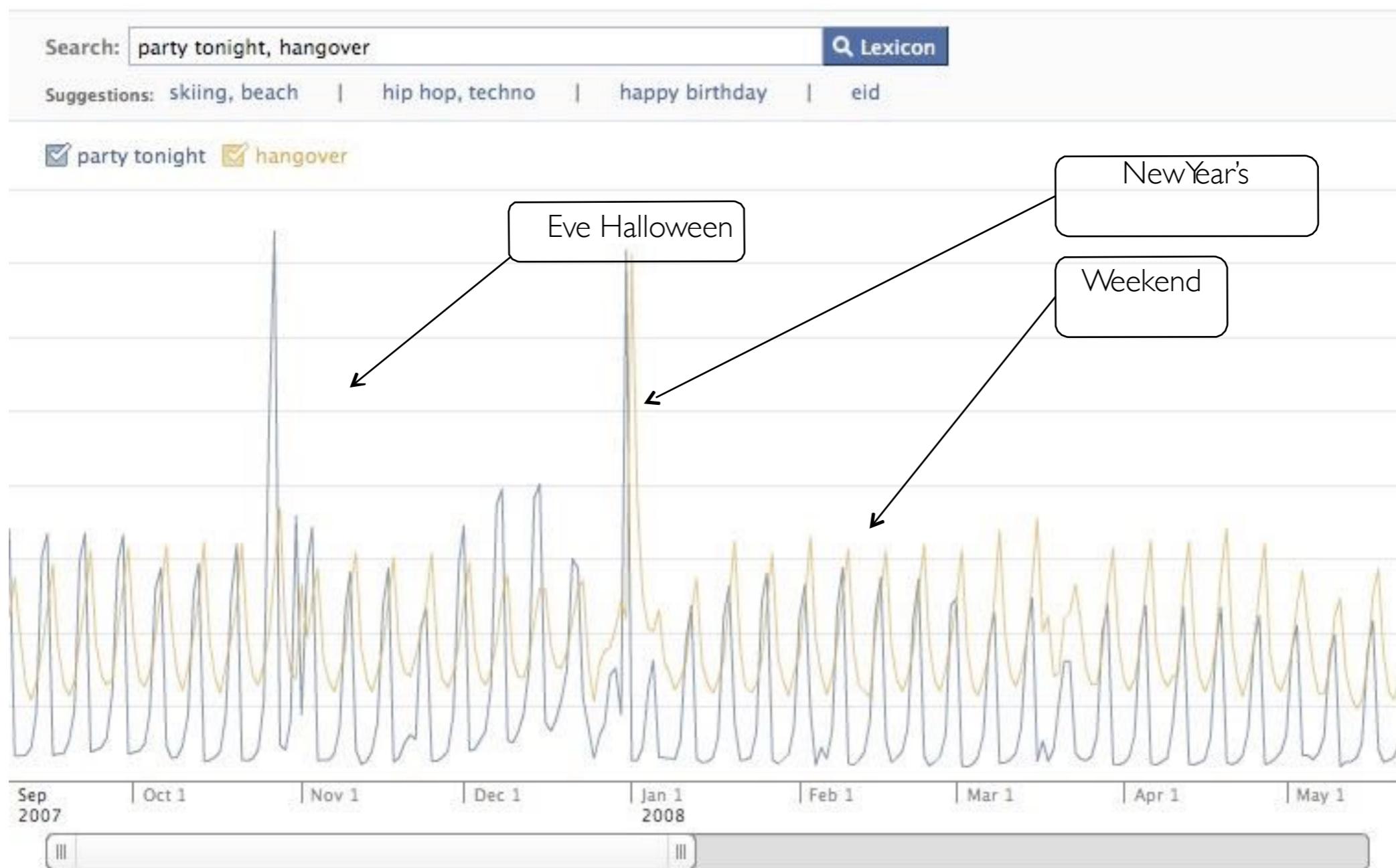
Data Science?

...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

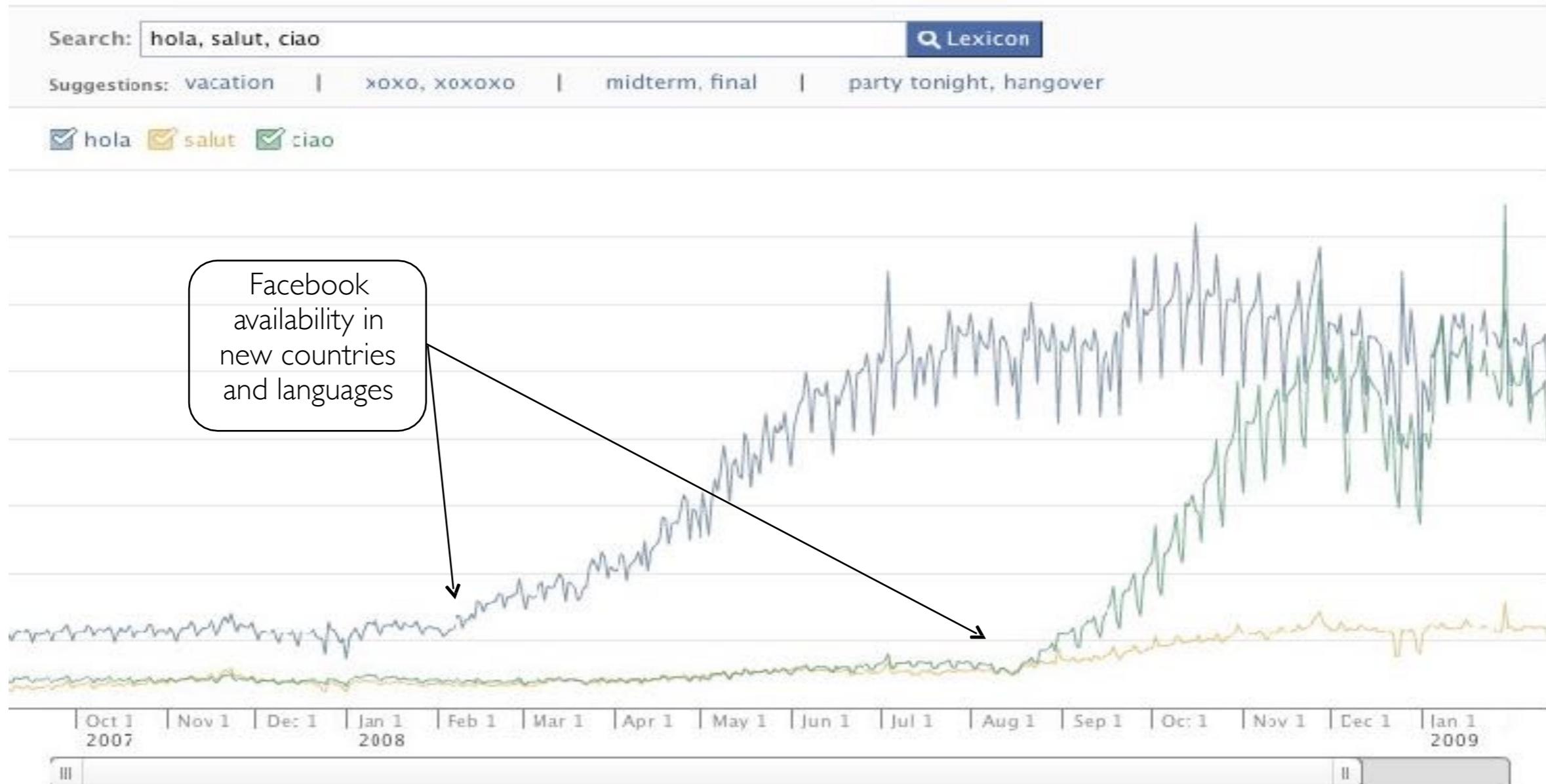
...that allowed the Obama campaign not only to alter the very nature of the electorate, making it younger and less white, but also to create a portrait of shifting voter allegiances. The power of this operation stunned Mr. Romney's aides on election night, as they saw voters they never even knew existed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

[NewYorkTimes](#),Wed Nov 7,2012

Data Science?



Data Science?



Data Science?

Epidemiological modeling of online social network dynamics

John Cannarella¹, Joshua A. Spechler^{1,*}

¹ Department of Mechanical and Aerospace Engineering, Princeton University, Princeton, NJ, USA

* E-mail: Corresponding spechler@princeton.edu

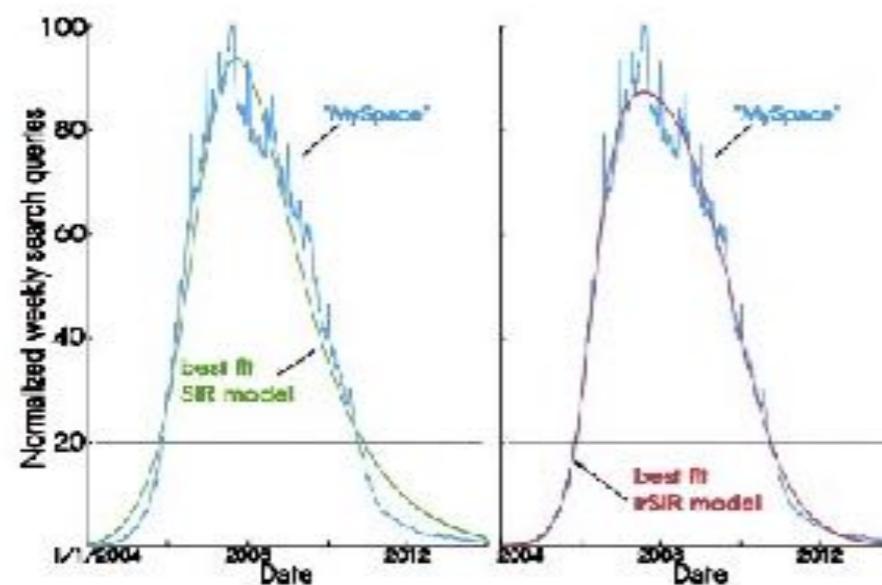
Abstract

The last decade has seen the rise of immense online social networks (OSNs) such as MySpace and Facebook. In this paper we use epidemiological models to explain user adoption and abandonment of OSNs, where adoption is analogous to infection and abandonment is analogous to recovery. We modify the traditional SIR model of disease spread by incorporating infectious recovery dynamics such that contact between a recovered and infected member of the population is required for recovery. The proposed infectious recovery SIR model (irSIR model) is validated using publicly available Google search query data for “MySpace” as a case study of an OSN that has exhibited both adoption and abandonment phases. The irSIR model is then applied to search query data for “Facebook,” which is just beginning to show the onset of an abandonment phase. Extrapolating the best fit model into the future predicts a rapid decline in Facebook activity in the next few years.

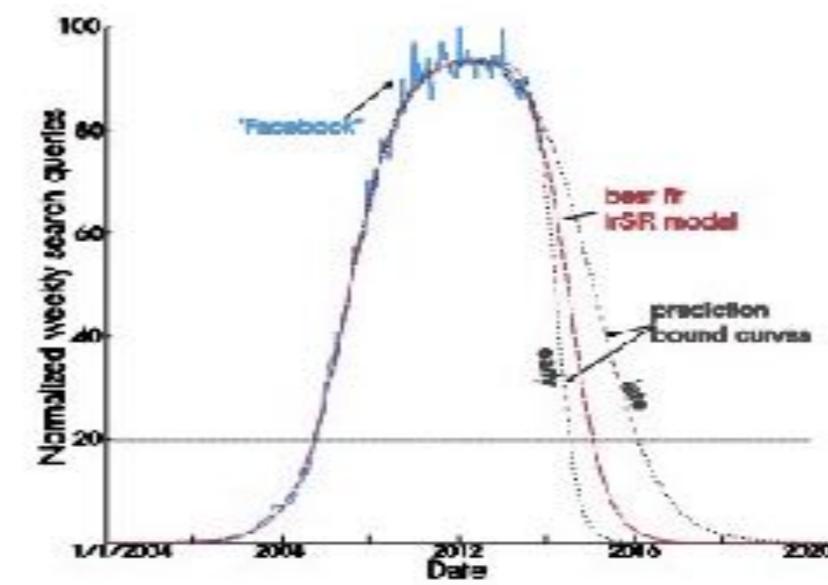
“Extrapolating the best fit model into the future predicts a rapid decline in Facebook activity in the next few years.”

Data Science?

Google Trends searches for
“MySpace”

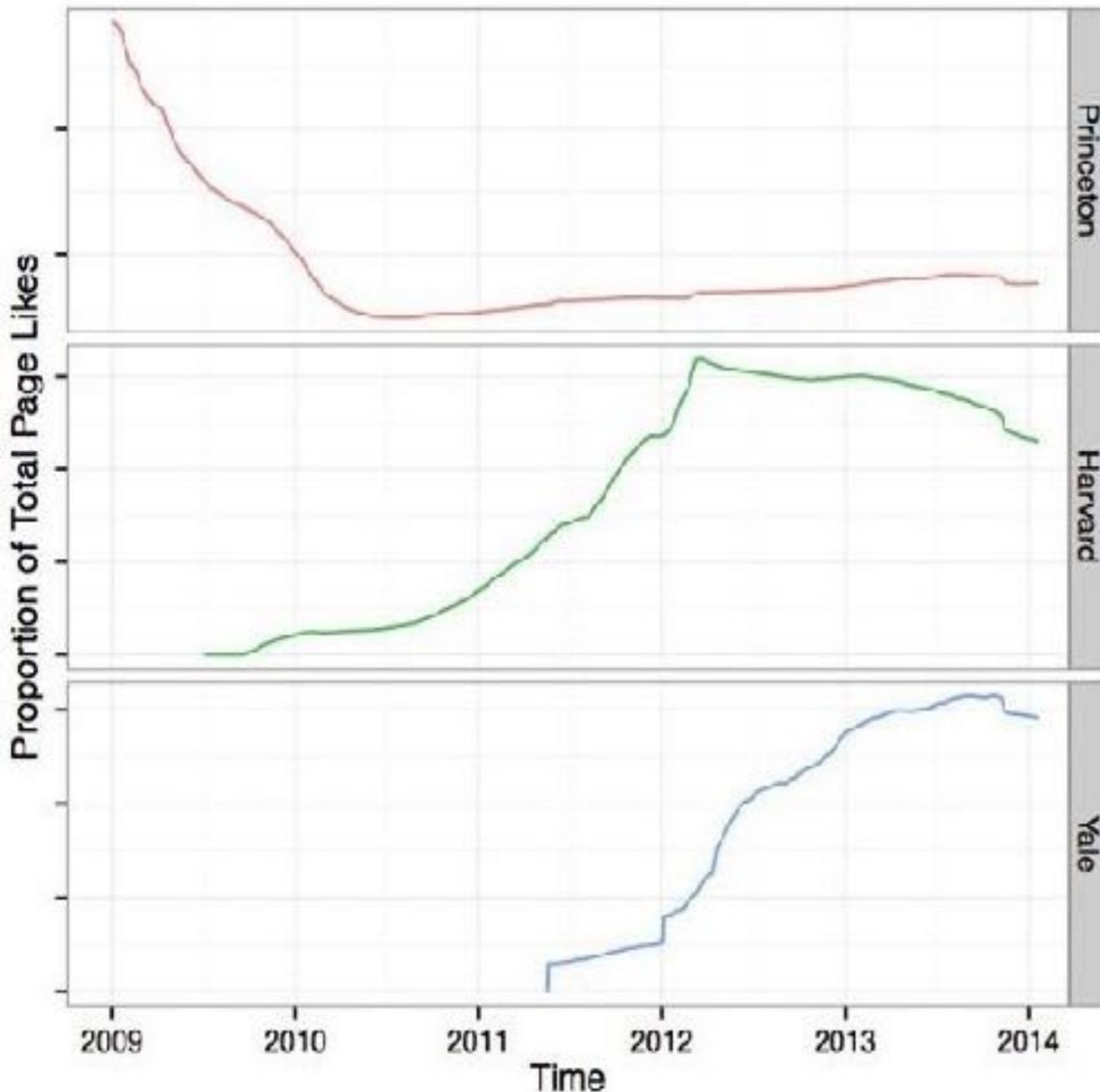


**Two Figures from
the paper**



Searches for
“Facebook”

Data Science?



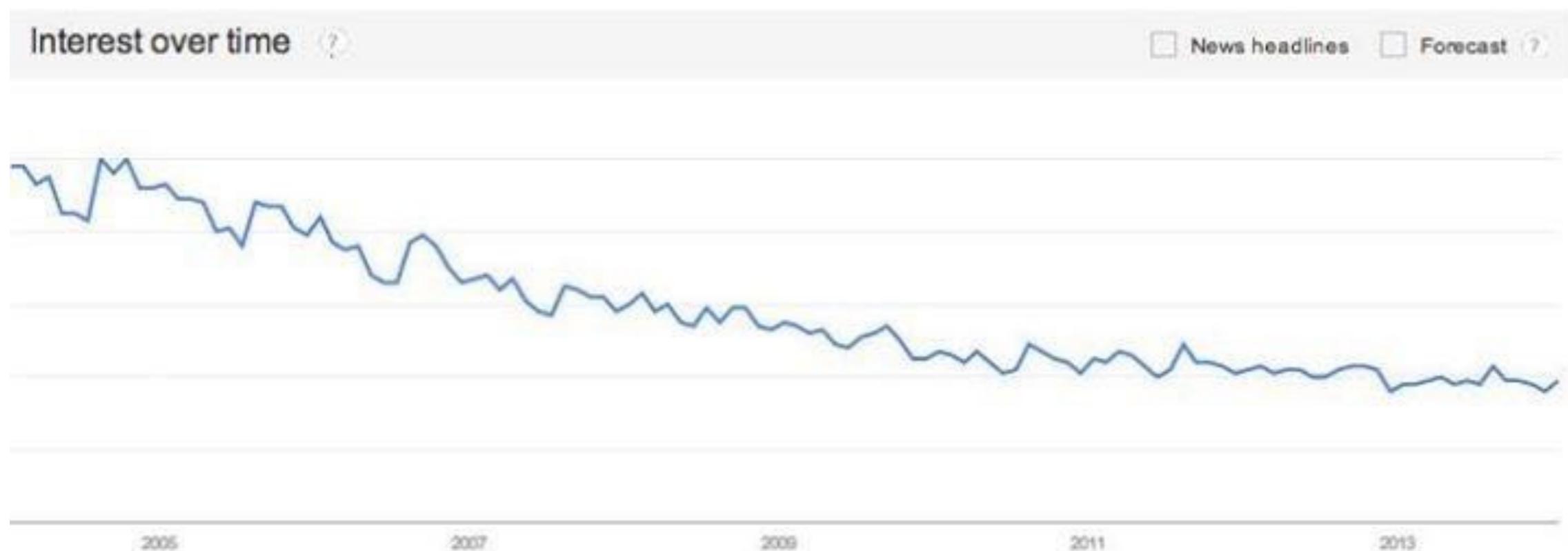
<https://www.facebook.com/notes/mike-develin/debunking-princeton/10151947421191849>

In keeping with the scientific principle "correlation equals causation**," our research unequivocally demonstrated that Princeton may be in danger of disappearing entirely.**

Data Science?

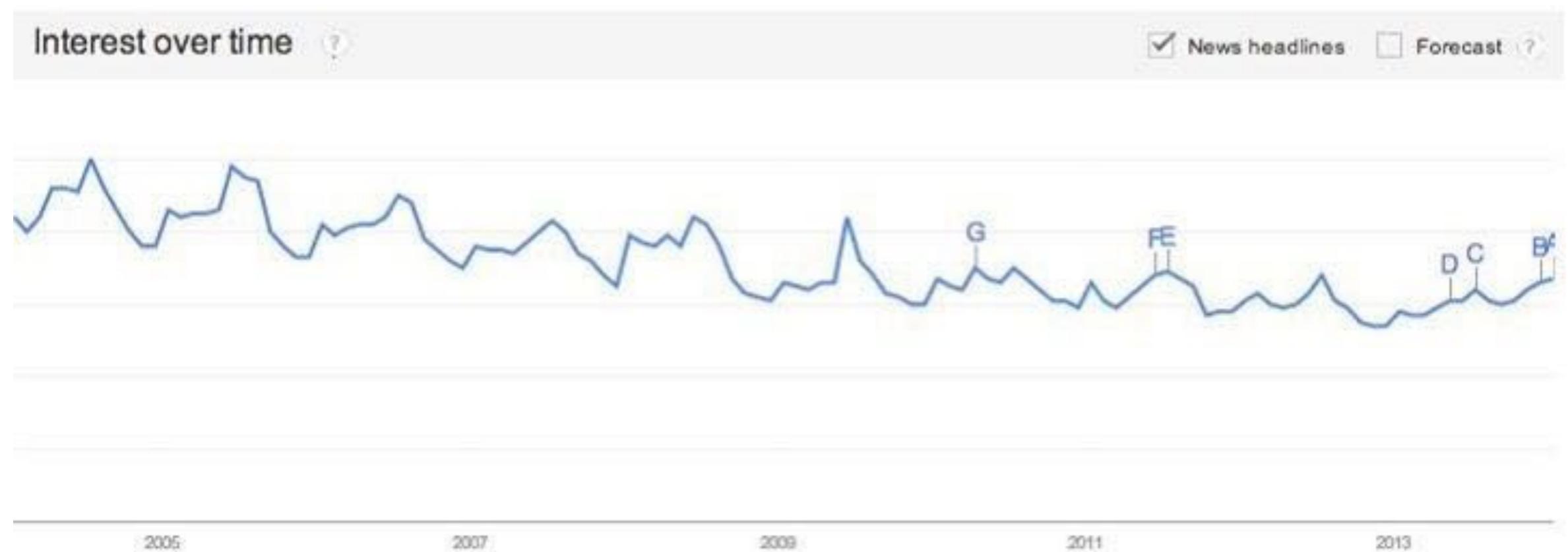
... and based on Princeton search trends:

“This trend suggests that Princeton will have only half its current enrollment by 2018, and by 2021 it will have no students at all,...”



Data Science?

While we are concerned for Princeton University, we are even more concerned about the fate of the planet — Google Trends for “air” have also been declining steadily, and our projections show that by the year 2060 there will be no air left:



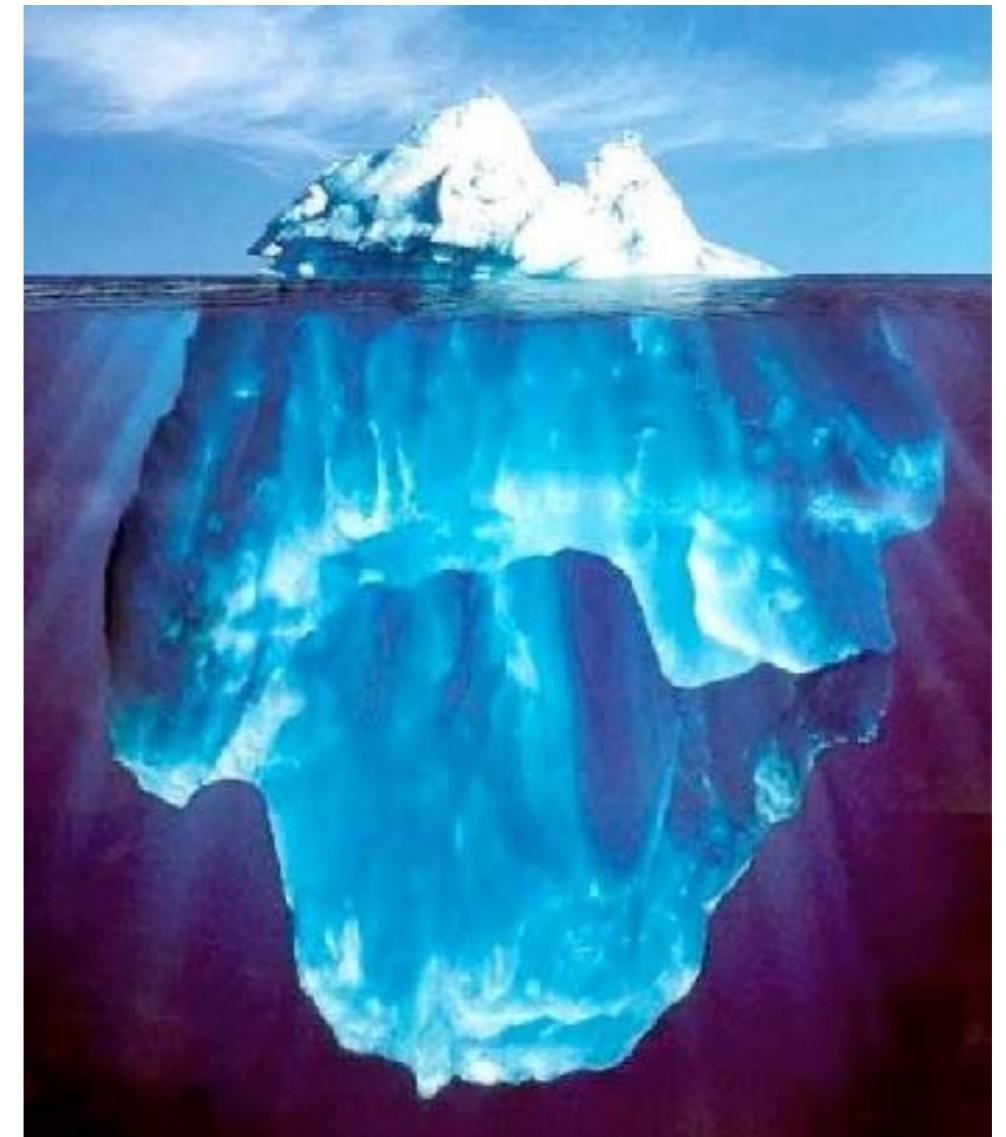


Technology behind Data Science

Fundamental Data Science for Data Scientist

Where Does Big Data Come From?

- It's all happening online – could record every:
 - » Click
 - » Ad impression
 - » Billing event
 - » Fast Forward, pause,...
 - » Server request
 - » Transaction
 - » Network message
 - » Fault
 - » ...

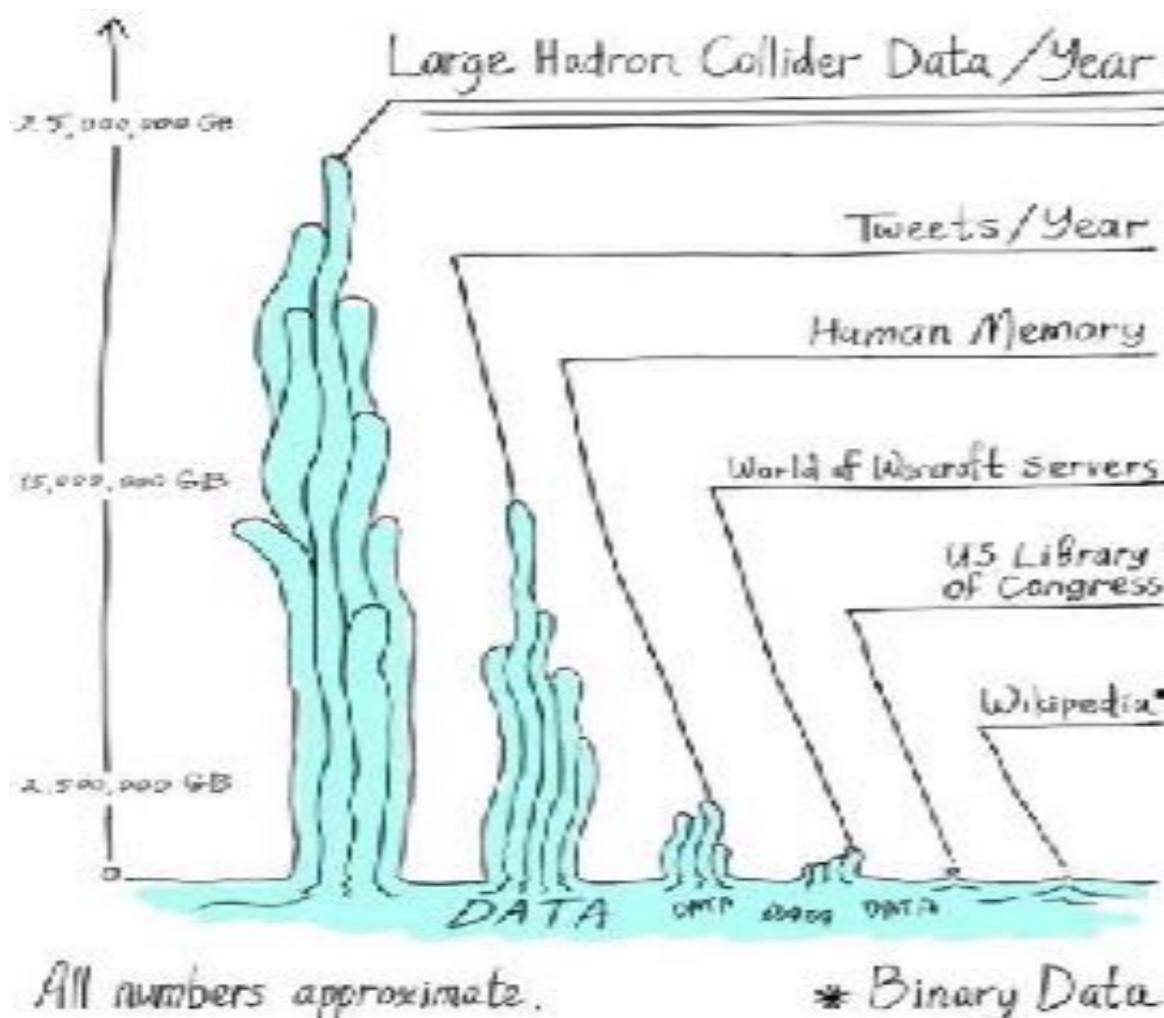


Where Does Big Data Come From?

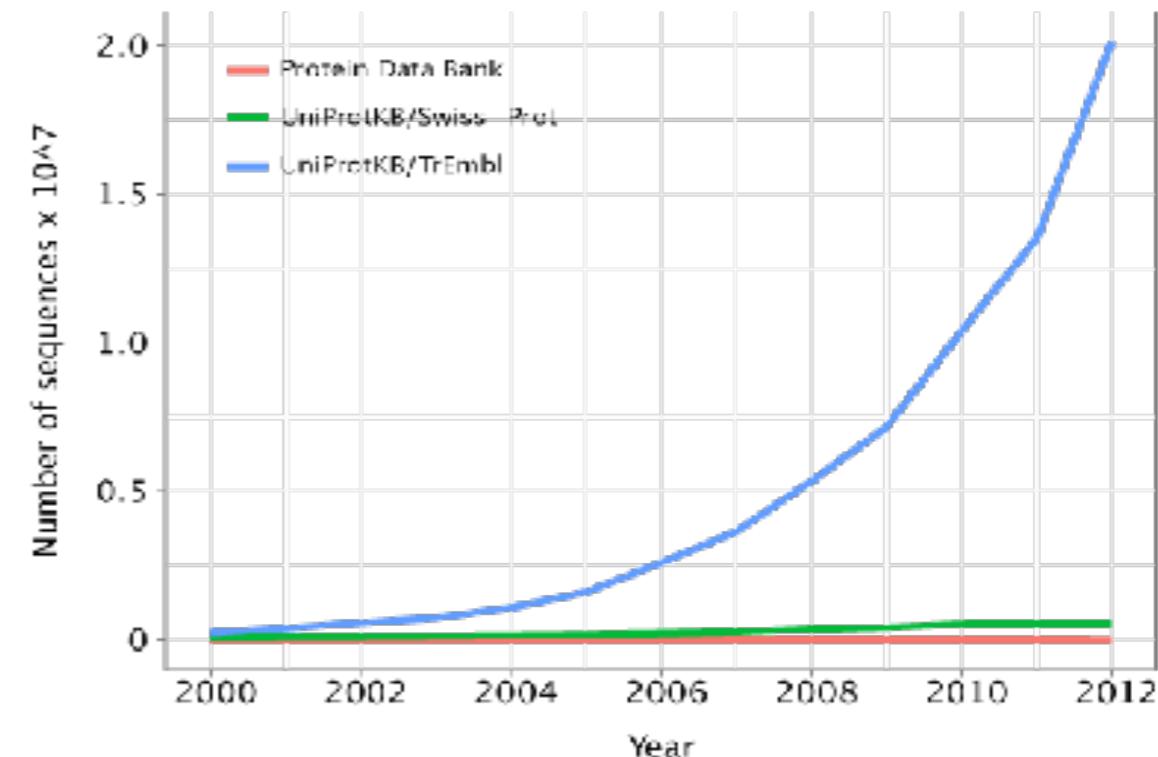
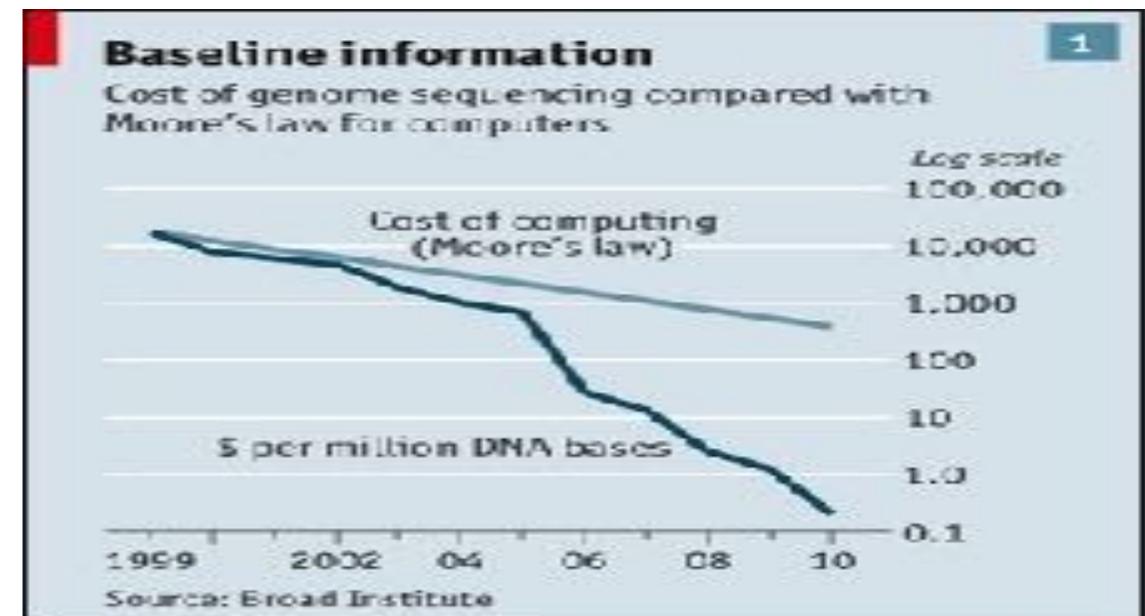
- User Generated Content (Web & Mobile)
 - » Facebook
 - » Instagram
 - » Yelp
 - » TripAdvisor
 - » Twitter
 - » YouTube
 - » ...

Where Does Big Data Come From?

- Health and Scientific Computing



Images: <http://www.economist.com/node/16349358> <http://gorbi.irb.hr/en/method/growth-of-sequence-databases/> <http://www.symmetrymagazine.org/article/august-2012/particle-physics-tames-big-data>

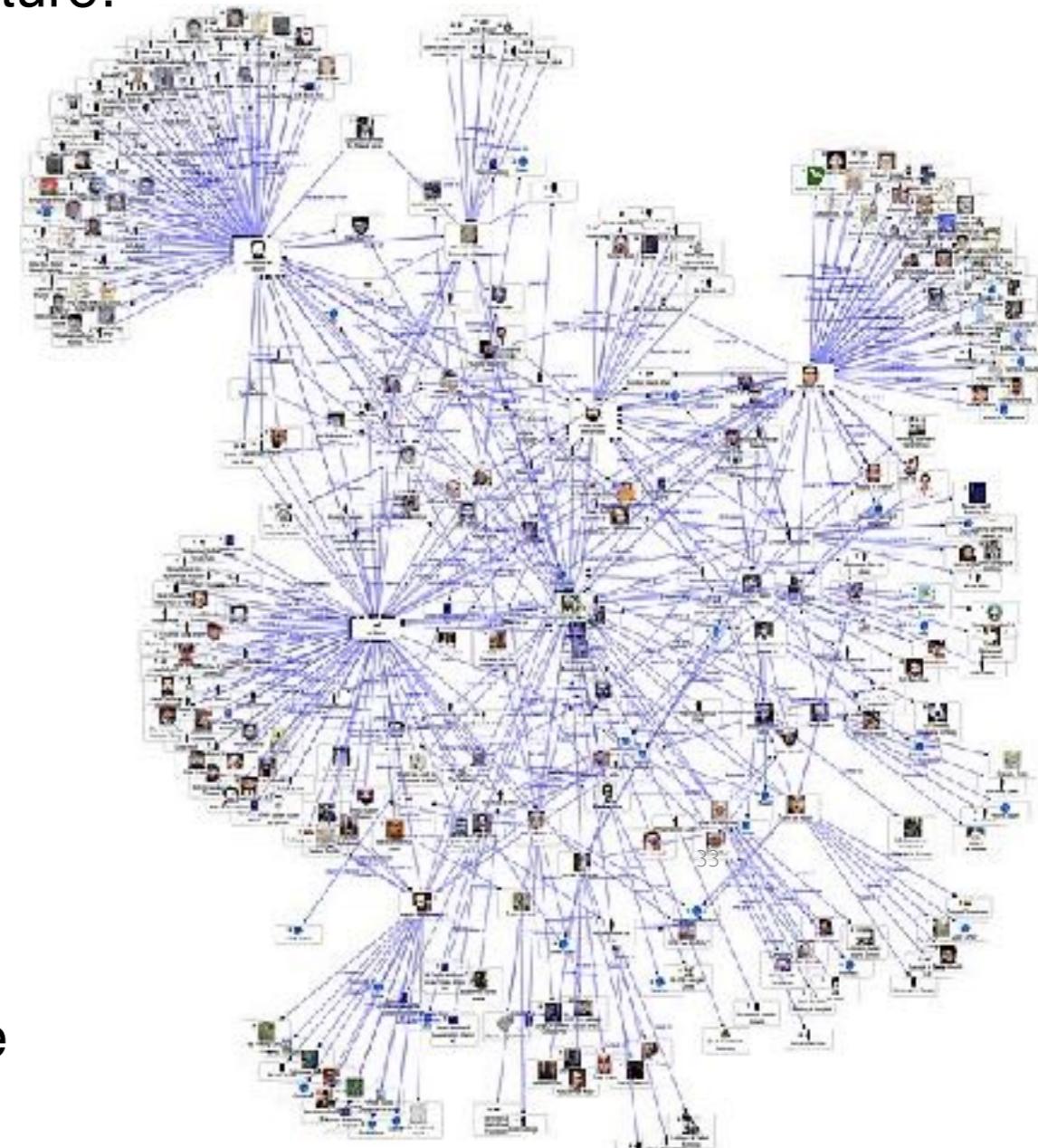


Graph Data

Lots of interesting data has a graph structure:

- Social networks
- Telecommunication Networks
- Computer Networks
- Road networks
- Collaborations/Relationships
- ...

Some of these graphs can get quite large
(e.g., Facebook user graph)



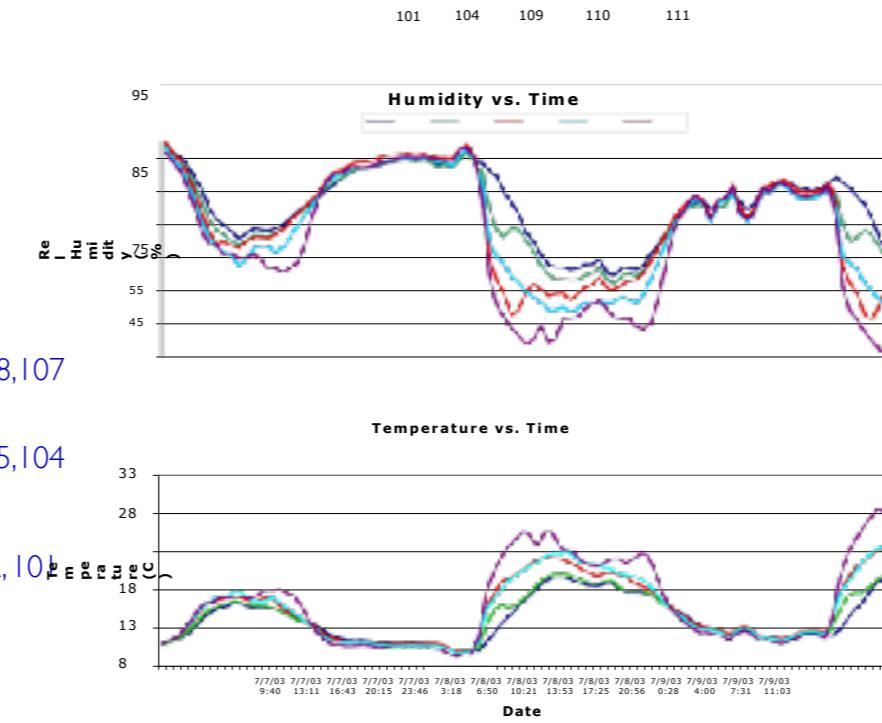
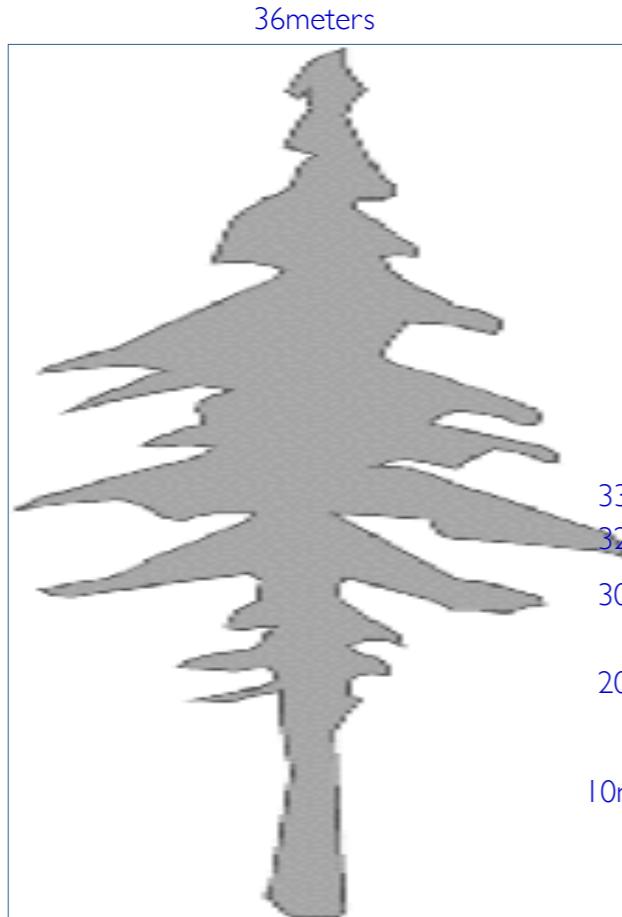
Log Files – Apache Web Server Log

```
uplherc.upl.com . . [01/Aug/1995:00:00:07 .0400] "GET / HTTP/1.0" 304 0
uplherc.upl.com . . [01/Aug/1995:00:00:08 .0400] "GET /images/ksclogo.medium.gif
HTTP/1.0" 304 0
uplherc.upl.com . . [01/Aug/1995:00:00:08 .0400] "GET /images/MOSAIC.logosmall.gif
HTTP/1.0" 304 0
uplherc.upl.com . . [01/Aug/1995:00:00:08 .0400] "GET /images/USA.logosmall.gif HTTP/
1.0" 304 0
ix.esc.ca2.07.ix.netcom.com . . [01/Aug/1995:00:00:09 .0400] "GET /images/launch.
logo.gif HTTP/1.0" 200 1713
uplherc.upl.com . . [01/Aug/1995:00:00:10 .0400] "GET /images/WORLD.logosmall.gif
HTTP/1.0" 304 0
slppp6.intermind.net . . [01/Aug/1995:00:00:10 .0400] "GET /history/skylab/
skylab.html HTTP/1.0" 200 1687
piweba4y.prodigy.com . . [01/Aug/1995:00:00:10 .0400] "GET /images/launchmedium.gif
HTTP/1.0" 200 11853
tampico.usc.edu . . [14/Aug/1995:22:57:13 .0400] "GET /welcome.html HTTP/1.0" 200 790
```

Machine Syslog File

```
dhcp.47.129:CS100_1> syslog .w 10
Feb 3 15:18:11 dhcp.47.129 Evernote[1140] <Warning>: .[EDAMAccounting read:]: unexpected
field ID 23 with type 8. Skipping.
Feb 3 15:18:11 dhcp.47.129 Evernote[1140] <Warning>: .[EDAMUser read:]: Skippi
Feb 3 15:18:11 dhcp.47.129 Evernote[1140] <Warning>: . [EDAMAuthenticationResult read:]: un
expected field ID 6 with type 11. Skipping.
Feb 3 15:18:11 dhcp.47.129 Evernote[1140] <Warning>: . [EDAMAuthenticationResult read:]: un
expected field ID 7 with type 11. Skipping.
Feb 3 15:18:11 dhcp.47.129 Evernote[1140] <Warning>: .[EDAMAccounting read:]: unexpected
field ID 19 with type 8. Skipping.
Feb 3 15:18:11 dhcp.47.129 Evernote[1140] <Warning>: .[EDAMAccounting
read:]: unexpected field ID 23 with type 8. Skipping.
Feb 3 15:18:11 dhcp.47.129 Evernote[1140] <Warning>: .[EDAMUser read:]: un
expected field ID 17 with type 12. Skipping.
Fe 3 15:18:11 dhcp.47.129 Evernote[1140] <Warning>: .[EDAMSyncState read:]: b
unexpected field ID 5 with type 10. Skipping.
Feb 3 15:18:49 dhcp.47.129 com.apple.mtmd[47] <Notice>: low priority
thinning needed for volume Macintosh HD (/) with 18.9 <= 20.0 pct free space
```

Internet of Thing



Redwood tree humidity and temperature at various heights

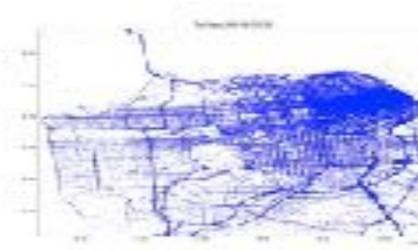
Internet of Things: RFID tags

- California FasTrak Electronic Toll Collection transponder
- Used to pay tolls
- Collected data also used for traffic reporting
 - » <http://www.511.org/>



<http://en.wikipedia.org/wiki/FasTrak>

What Can You do with Big Data?



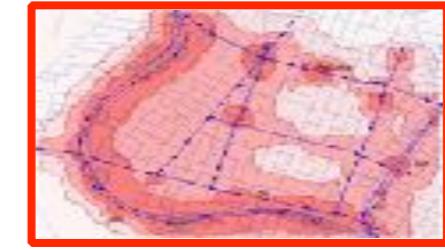
Crowdsourcing



+ Physical modeling



+ Sensing



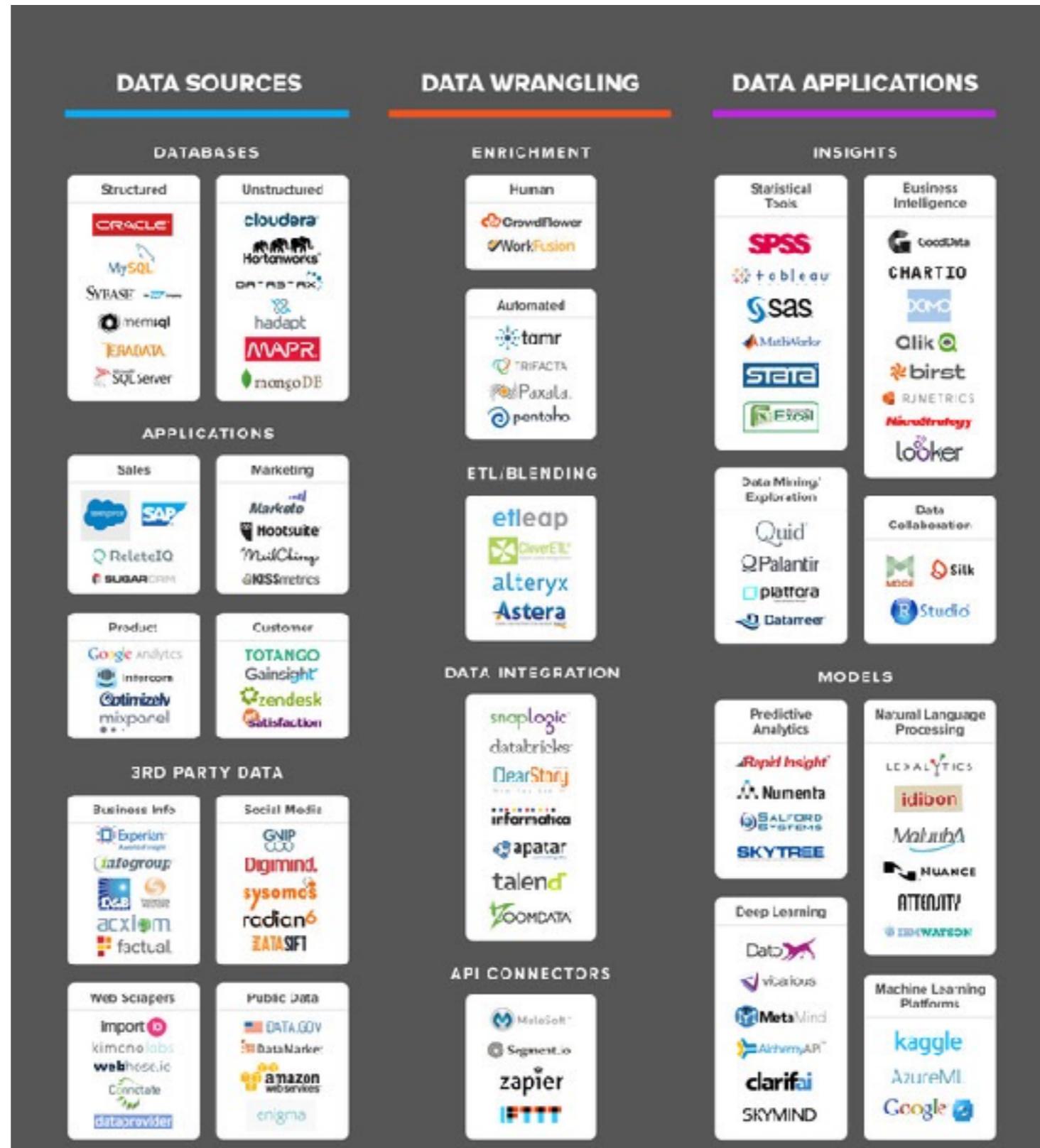
+ Data Assimilation

=



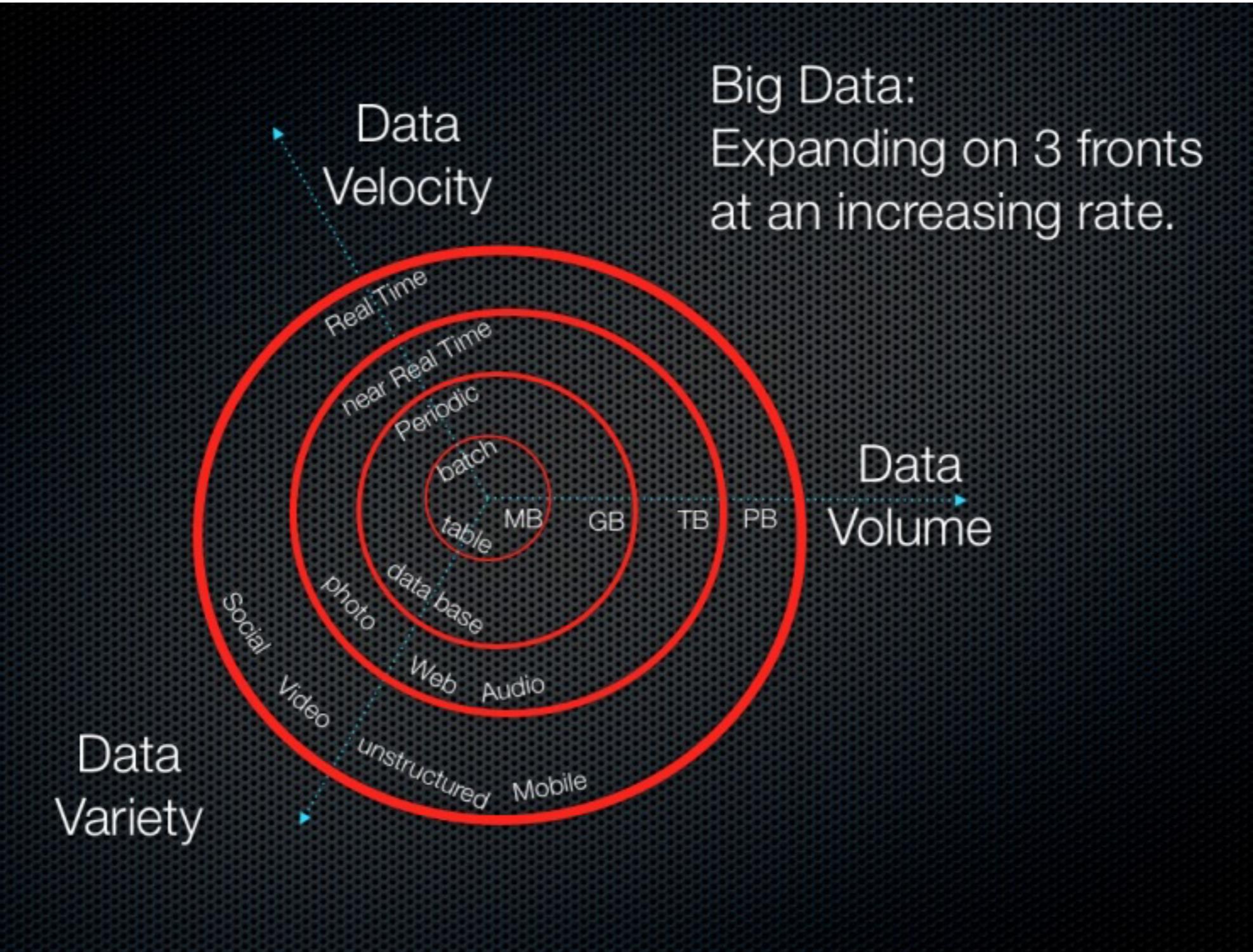
<http://traffic.berkeley.edu>

Tools for Data Science Projects



Big Data

Big Data:
Expanding on 3 fronts
at an increasing rate.



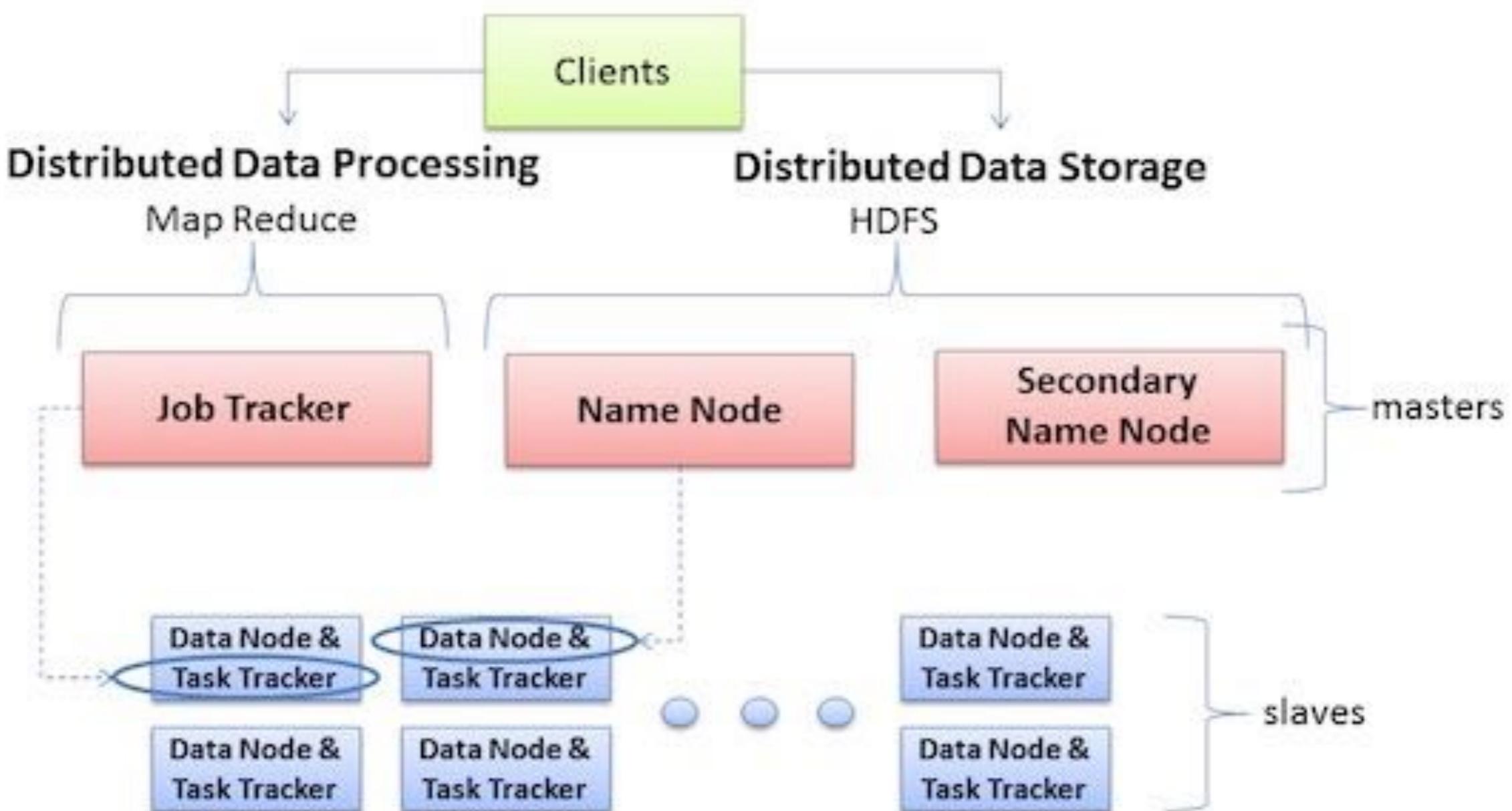
Hadoop

At the most basic level, Hadoop is an **open-source** software platform designed to store and process quantities of data that are too large for just one particular device or server.

Hadoop's strength lies in its ability to **scale across thousands of commodity servers that don't share memory or disk space.**



How Hadoop works





The Roles of a Data Scientist

Fundamental Data Science for Data Scientist

The role of Data Scientist

- The many roles that data scientists can play fall into the following domains.
 - Decision sciences and business intelligence
 - Product and marketing analytics
 - Fraud, abuse, risk and security
 - Data services and operations
 - Data engineering and infrastructure
 - Organizational and reporting alignment

Characteristics of Data Scientist

- **Curiosity** is necessary to peel apart a problem and examine the interrelationships between data that may appear superficially unrelated.
- **Creativity** is required to invent and try new approaches to solving a problem, which often times have never been applied in such a context before.
- Focus is required to design and test a technique over days and weeks, find it doesn't work, learn from the failure, and try again.
- **Attention to Detail** is needed to maintain rigor, and to detect and avoid over-reliance on intuition when examining data.

Train your Data Scientist

- Finding rich data sources.
- Working with large volumes of data despite hardware, software, and bandwidth constraints.
- Cleaning the data and making sure that data is consistent.
- Melding multiple datasets together.
- Visualizing that data.

Hiring a unicorn?

MODERN DATA SCIENTIST

Data Scientist, the sexiest job of 21st century requires a mixture of multidisciplinary skills ranging from an intersection of mathematics, statistics, computer science, communication and business. Finding a data scientist is hard. Finding people who understand who a data scientist is, is equally hard. So here is a little cheat sheet on who the modern data scientist really is.



- MATH & STATISTICS**
 - ★ Machine learning
 - ★ Statistical modeling
 - ★ Experiment design
 - ★ Bayesian inference
 - ★ Supervised learning: decision trees, random forests, logistic regression
 - ★ Unsupervised learning: clustering, dimensionality reduction
 - ★ Optimization: gradient descent and variants
- PROGRAMMING & DATABASE**
 - ★ Computer science fundamentals
 - ★ Scripting language e.g. Python
 - ★ Statistical computing package e.g. R
 - ★ Databases SQL and NoSQL
 - ★ Relational algebra
 - ★ Parallel databases and parallel query processing
 - ★ MapReduce concepts
 - ★ Hadoop and Hive/Fig
 - ★ Custom reducers
 - ★ Experience withaaS like AWS
- DOMAIN KNOWLEDGE & SOFT SKILLS**
 - ★ Passionate about the business
 - ★ Curious about data
 - ★ Influence without authority
 - ★ Hacker mindset
 - ★ Problem solver
 - ★ Strategic, proactive, creative, innovative and collaborative
- COMMUNICATION & VISUALIZATION**
 - ★ Able to engage with senior management
 - ★ Story telling skills
 - ★ Translate data-driven insights into decisions and actions
 - ★ Visual art design
 - ★ R packages like ggplot or lattice
 - ★ Knowledge of any of visualization tools e.g. Flare, D3.js, Tableau

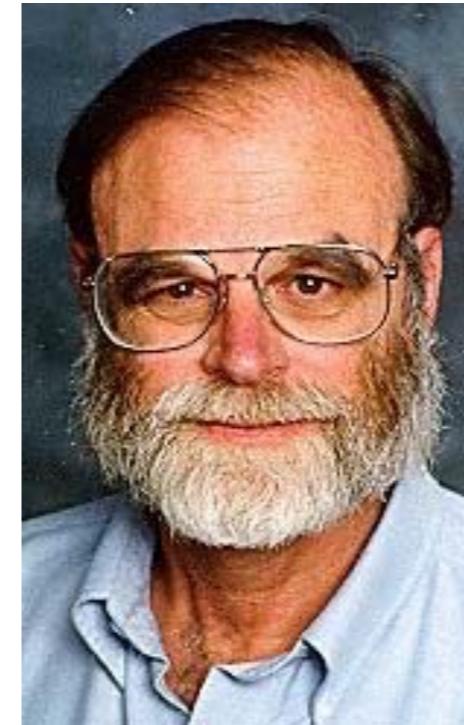


Data Science Life Cycle

Introduction to Data Science

Jim Gray's Model

- Capture
- Curate
- Communicate



Turing award winner

Ben Fry's Model

- Acquire
- Parse
- Filter
- Mine
- Represent
- Refine
- Interact



Data visualization expert

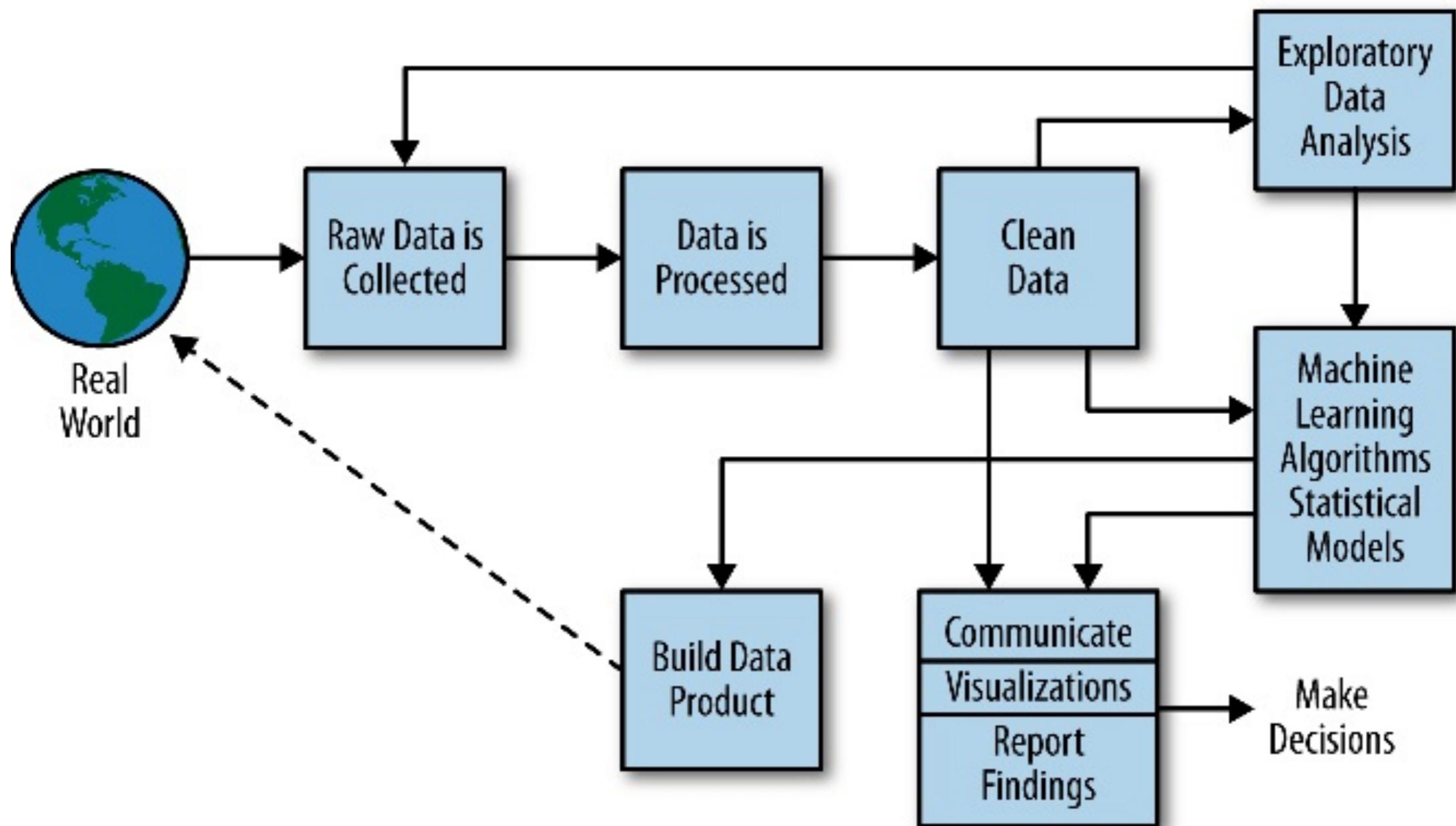
Jeff Hammerbacher's Model

1. Identify problem
2. Instrument data sources
3. Collect data
4. Prepare data (integrate, transform,
clean, filter, aggregate)
5. Build model
6. Evaluate model
7. Communicate results



Facebook, Cloudera

Data Science Process

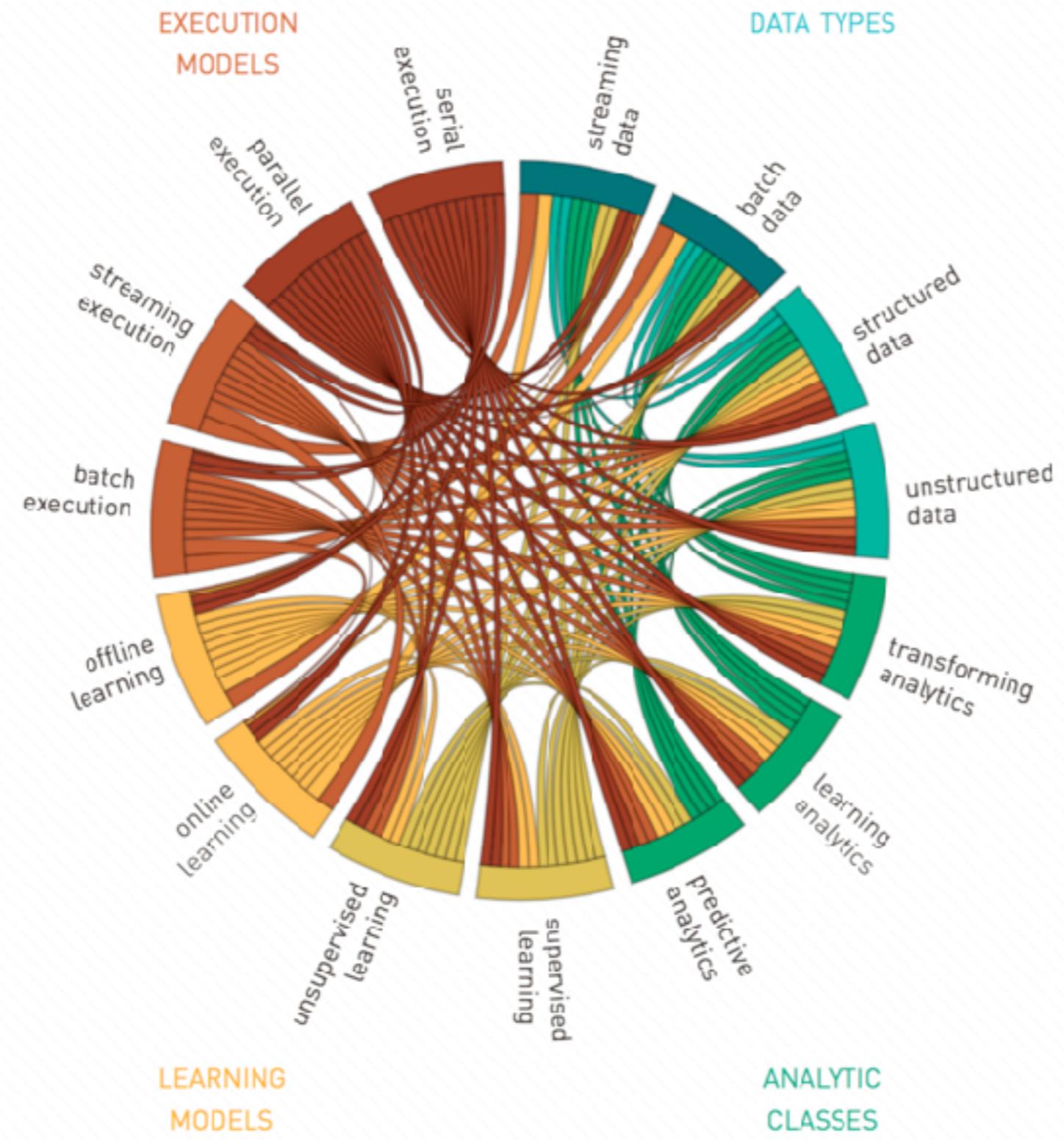




Component of Data Science

Fundamental Data Science for Data Scientist

Components of Data Science



Data Types

- Structured vs. Unstructured

Structured

- Categorical
 - Ordinal
 - Nominal

- Ratio
- Continuous

Unstructured

- Images
- Video
- Audio

Data Types

- Streaming vs. Batching
 - Streaming : Twitter stream, Webpage click stream
 - Batching : ERP data, CRM data

Classes of Analytic Techniques



Transform Analytics

- **Aggregation:** Techniques to summarize the data. These include basic statistics (e.g., mean, standard deviation), distribution fitting, and graphical plotting.
- **Enrichment:** Techniques for adding additional information to the data, such as source information or other labels.
- **Processing:** Techniques that address data cleaning, preparation, and separation. This group also includes common algorithm pre-processing activities such as transformations and feature extraction.

Learning Analytics

- **Regression:** Techniques for estimating relationships among variables, including understanding which variables are important in predicting future values.
- **Clustering:** Techniques to segment the data into naturally similar groups.
- **Classification:** Techniques to identify data element group membership.
- **Recommendation:** Techniques to predict the rating or preference for a new entity, based on historic preference or behavior.

Predictive Analytics

- **Simulation:** Techniques to imitate the operation of a real-world process or system. These are useful for predicting behavior under new conditions.
- **Optimization:** Operations Research techniques focused on selecting the best element from a set of available alternatives to maximize a utility function.

Learning Model



Type of Algorithm

- Clustering
- Association learning
- Parameter estimation
- Recommendation engines
- Classification
- Similarity matching
- Neural networks
- Bayesian networks
- Genetic algorithms



Introduction to R

Fundamental Data Science for Scientist

What is R?

R is a system for statistical computation and graphics.

- It is heavily influenced by the S language
- R was initially written by Ross Ihaka and Robert Gentleman at the Department of Statistics of the University of Auckland in Auckland, New Zealand.
- The “R Core Team” maintain the source code for the software and release regular updates

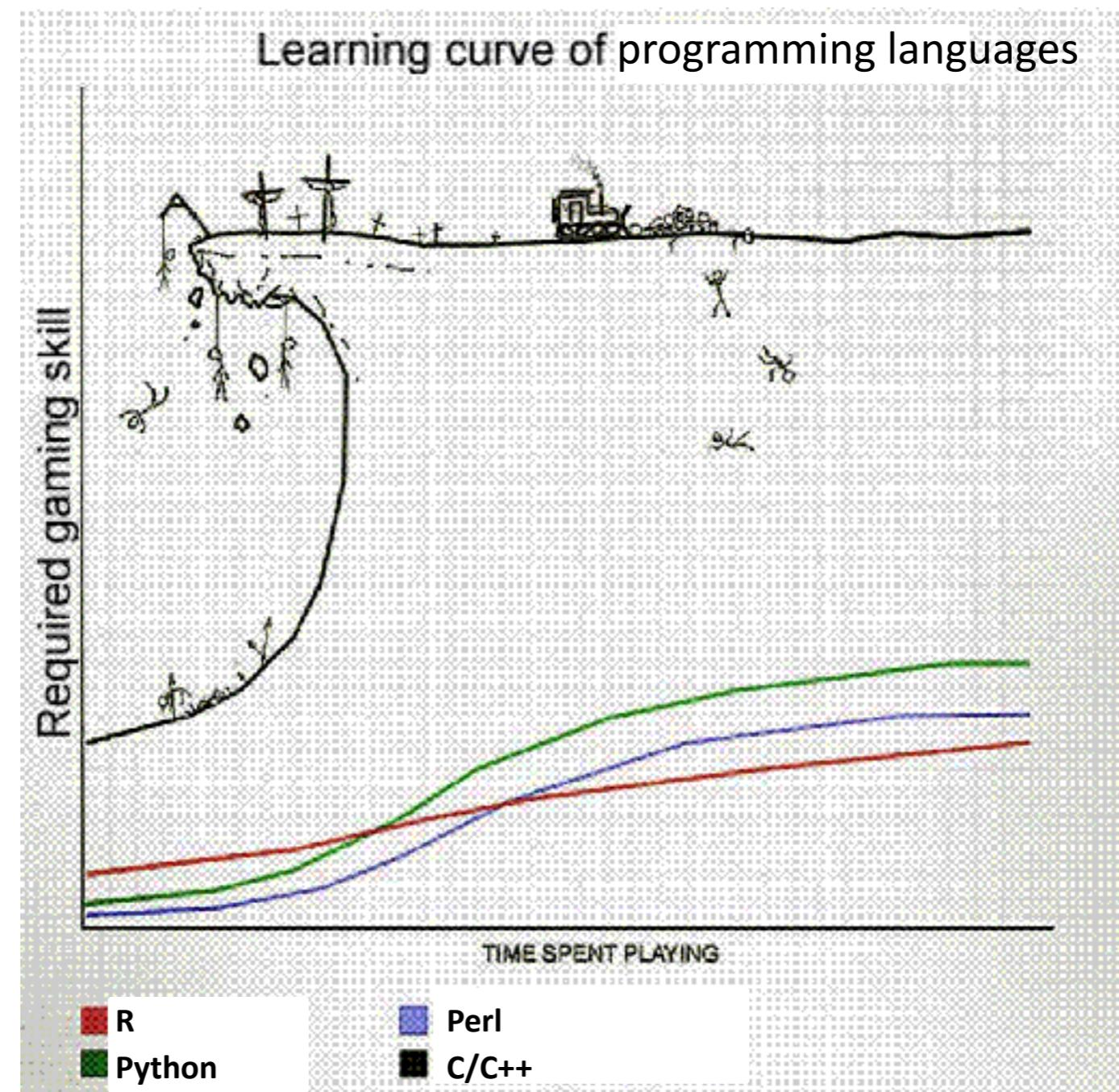
What is R?

- In addition, the R project is added to by many of its users, who write source code for many different types of analytical procedures
- Everything from analytical chemistry to epidemiology to linguistics
 - Currently 8,117 different user--written libraries available

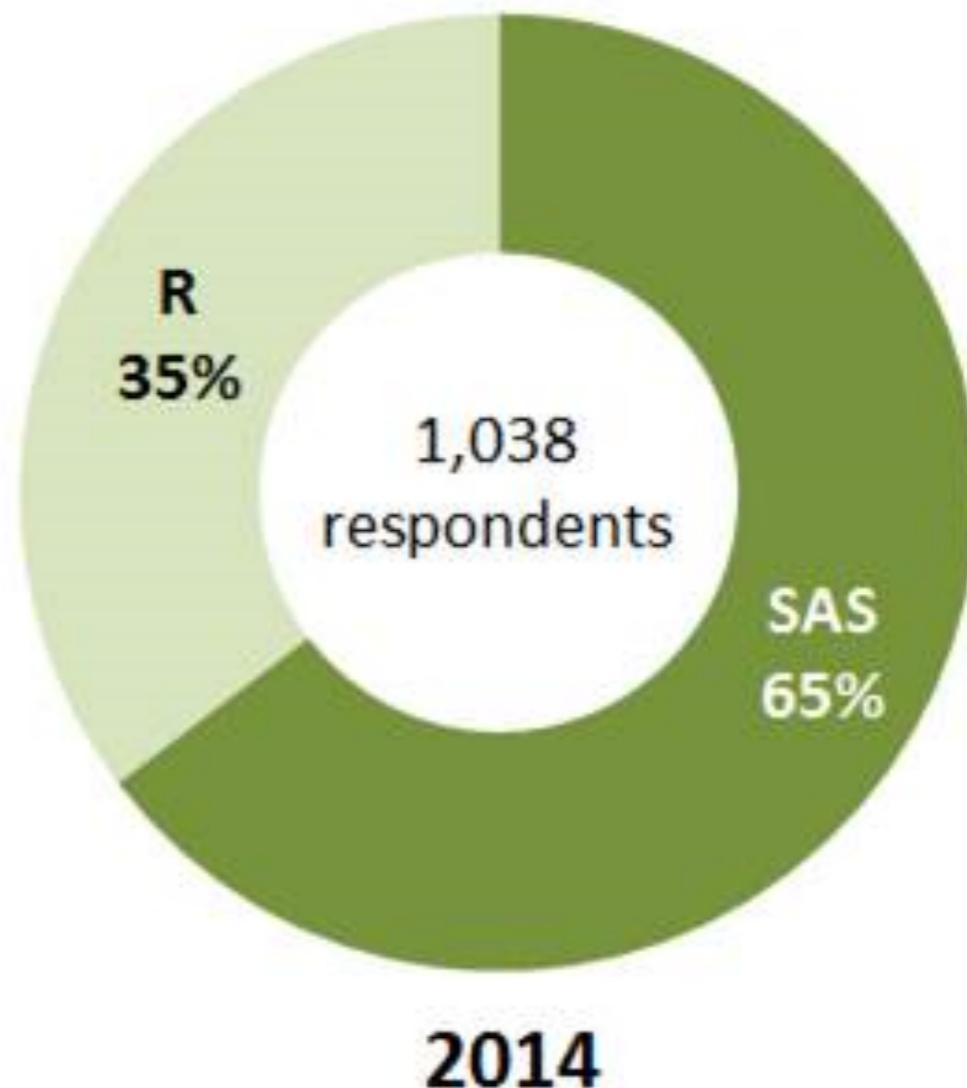
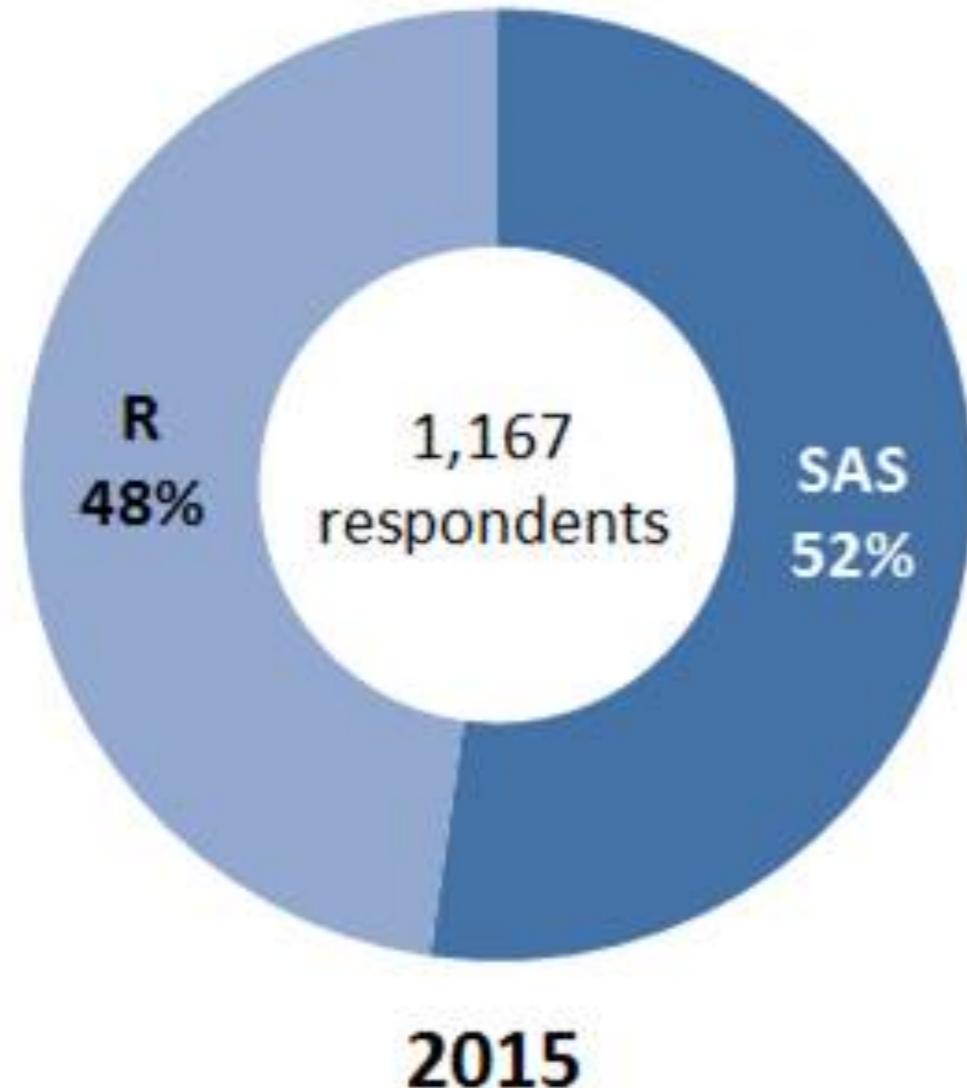
Why use R?

- R is Open-Source Software
- Many built-in functions and installable packages that will cover nearly every possible need
- R is an interpreted language
 - Code doesn't have to be compiled
- Interactive console makes testing and debugging easy
- Cons to using R
 - Slower than compiled languages
 - Can have runtime errors

Why use R?

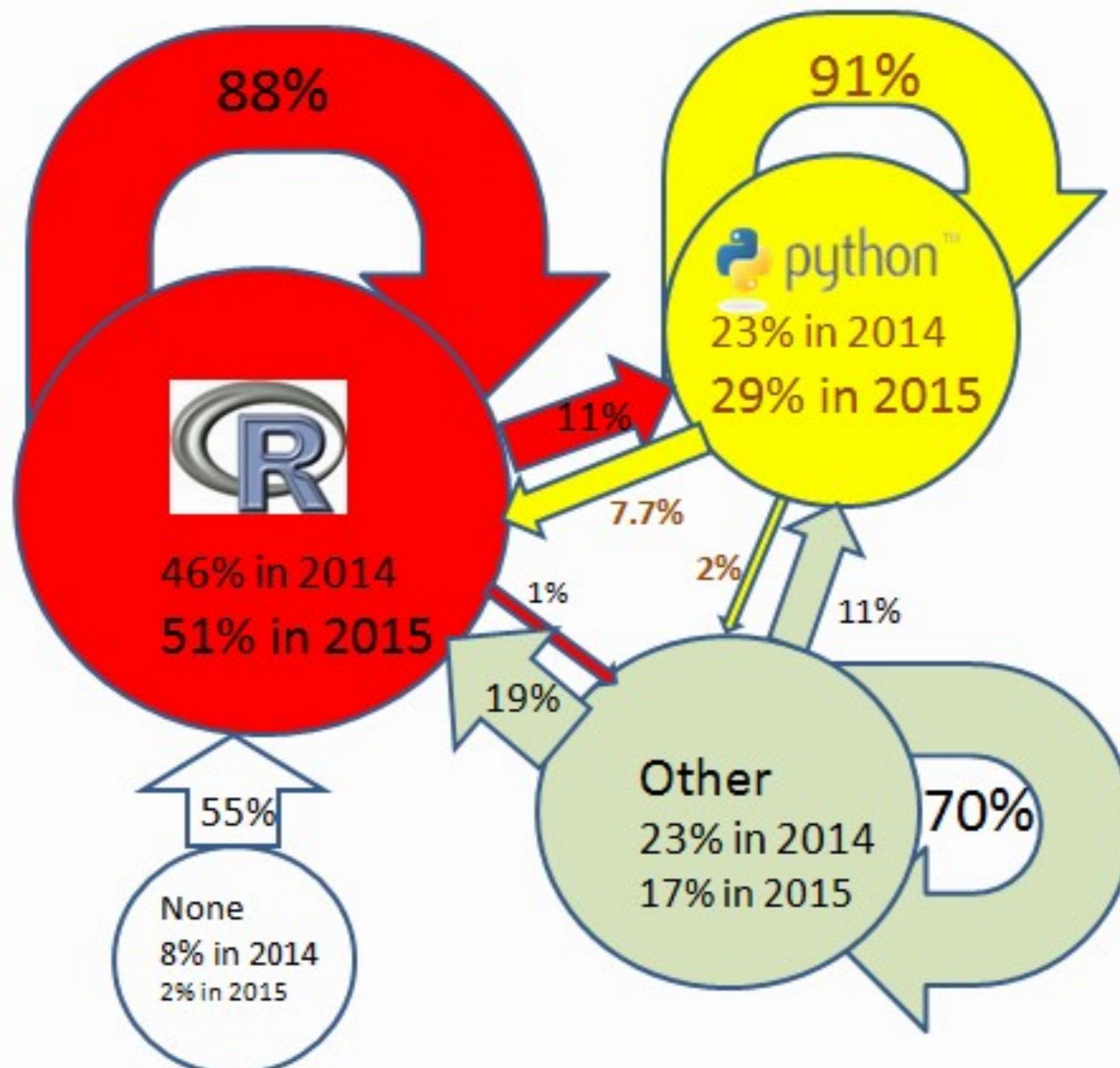


R vs SAS



R vs Python

Primary Analytics, Data Mining, Data Science
Languages in 2014 vs 2015



Tools



R
www.r-project.org

The engine*



Rstudio
www.rstudio.org

The pretty face**

* Many alternatives exist. Smallest learning curve.

** A few alternatives exist. This happens to be the easiest at the moment.

Installation

Install R on Windows

<https://cran.r-project.org/bin/windows/base/>

Install R on Mac OSX

<https://cran.r-project.org/bin/macosx/>

Install R-Studio

<https://www.rstudio.com/products/rstudio/download/>

Introducing Rstudio

The screenshot shows the RStudio interface with several red annotations:

- console** (just like Rterm) - points to the dark blue central area.
- workspace: your variables** - points to the top right panel titled "Workspace".
- history: last command** - points to the bottom right panel titled "History".
- files** - points to the leftmost tab of the file browser.
- plots** - points to the second tab of the file browser.
- packages** - points to the third tab of the file browser.
- help** - points to the fourth tab of the file browser.

Console: r/f >

Workspace | **History**

File | **Plot** | **Packages** | **Help**

New Folder | Delete | Rename | Help

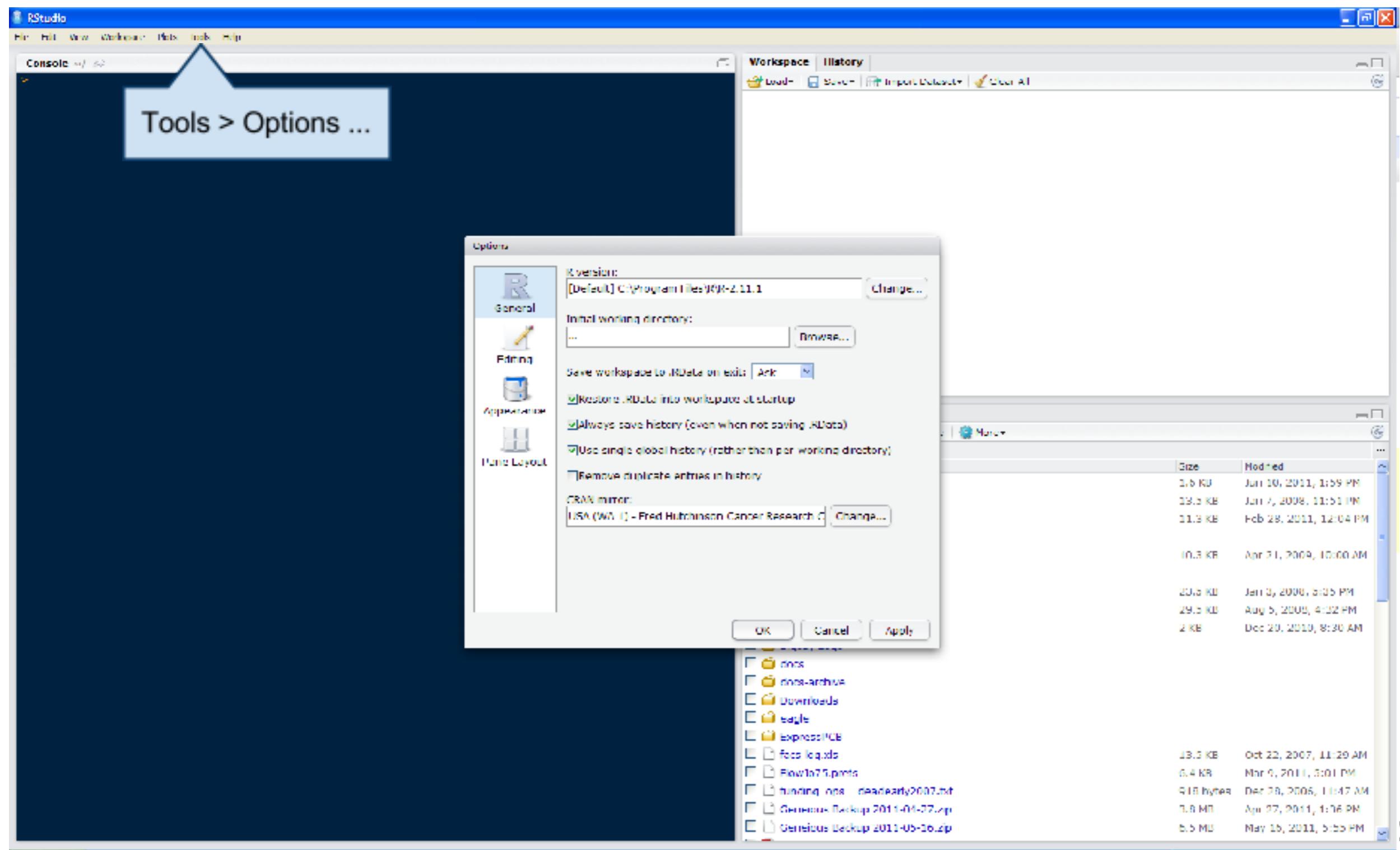
Name Size Modified

Name	Size	Modified
.Rhistory	1.0 KB	Jun 10, 2011, 1:39 PM
20070724_data.xls	13.5 KB	Jan 7, 2008, 11:51 PM
AutoHotkey.ahk	11.3 KB	Feb 28, 2011, 12:04 PM
bsk	10.3 KB	Apr 21, 2009, 10:00 AM
clientLib	21.5 KB	Jan 3, 2008, 5:35 PM
code	29.5 KB	Aug 5, 2008, 1:02 PM
counts.xls	2 KB	Dec 20, 2010, 8:03 AM
cuensim_seals		
reAnalHelp		
Digital Images		
dns		
dns-downloads		
Downloads		
eagle		
EPRESSOR		
first.lqxdn		
FlowIn75.pents		
funding_oos_deedonly2007.bct		
Gencious Backup 2011-04-27.zip	9.18 bytes	Dec 26, 2009, 11:47 AM
Gencious Backup 2011-05-16.zip	3.8 MB	Apr 27, 2011, 1:30 PM
	6.5 MB	May 10, 2011, 5:55 PM

Rstudio Features

- Code completion
- Command history search
- Command history to R script / file
- Function extraction from Rscript
- Sweave/Knitr support

Configuring Rstudio



Choosing a CRAN Mirror

- CRAN mirrors contain the R packages that can extend the functionality of R
- Choose a mirror located close to you as that will most likely give you the fastest downloads

Choosing Repositories

- Repositories host the packages
 - CRAN, CRANextra, BioCsoft, BioCann, BioCexp, BioCext, Omegahat, R-Forge and rforge.net
- Use this code to set your repositories

setRepositories()

```
> setRepositories()
--- Please select repositories for use in this session ---

1: + CRAN
2: + CRAN (extras)
3: BioC software
4: BioC annotation
5: BioC experiment
6: BioC extra
7: Omegahat
8: R-Forge
9: rforge.net

Enter one or more numbers separated by spaces, or an empty line to cancel
1:
```

Packages Tab

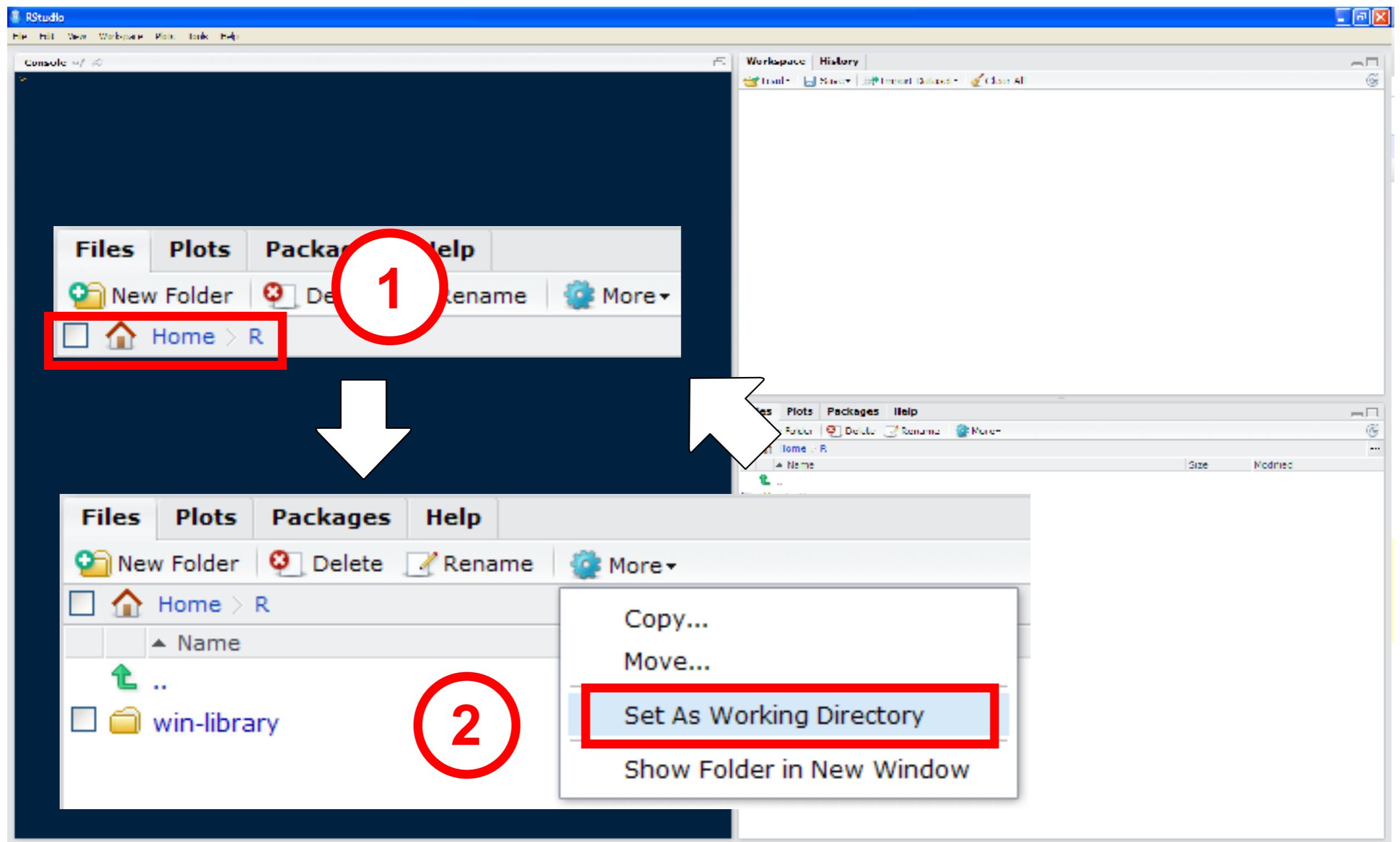
The screenshot shows the RStudio interface. The top menu bar includes File, Edit, Code, View, Project, Workspace, Plots, Tools, and Help. Below the menu is a toolbar with icons for file operations and a search bar labeled 'Go to function'. The main workspace contains a 'Console' tab showing R version 2.15.0 output, and a 'History' tab showing R code related to network analysis. A blue callout box labeled 'Packages Tab' points to the tab bar at the bottom of the window, which also includes File, Plots, Packages, and Help. The 'Packages' tab is currently selected, displaying a list of available packages. The list includes:

Package	Description
annotation	Annotation for microarrays
AnnotationDbi	Annotation Database Interface
Bioconductor	Basic Bioconductor functions for Bioconductor
BiocGenerics	Generic functions for Bioconductor
biomaRt	Interface to BioMart databases (e.g. Ensembl, COSMIC, Wormbase and Gtex)
bitops	Functions for Bitwise operations
boot	Bootstrap Functions (originally by Angelo Canty for S)
class	Functions for Classification
cluster	Cluster Analysis Extended Rousseeuw et al.
codetools	Code Analysis Toolkit for R
compiler	The R Compiler Package
coercion	Efficient Estimation of Covariance and (Partial) Correlation
data.table	The R DataTable Package
DBI	R Database Interface
fitclust	Flexible Mixture Modelling
foreign	Read Data Stored by Minitab, S, SAS, SPSS, Stata, Systat, dBase, ...
GO.db	A set of annotation maps describing the entire Gene Ontology
GOmin	Computation of functional similarities between GO terms and gene products by GO enrichment analysis
igraph	igraph: A package to handle graph data structures
graphics	The R Graphics Package
gridDevices	The R Graphics Devices and Support for Colours and Fonts

How do Packages Work?

- Packages tab lets you see what packages are installed
- A package must be loaded before you can use it
 - In Rstudio this is accomplished by clicking the checkbox next to the package name in the package tab
- We will be using different packages

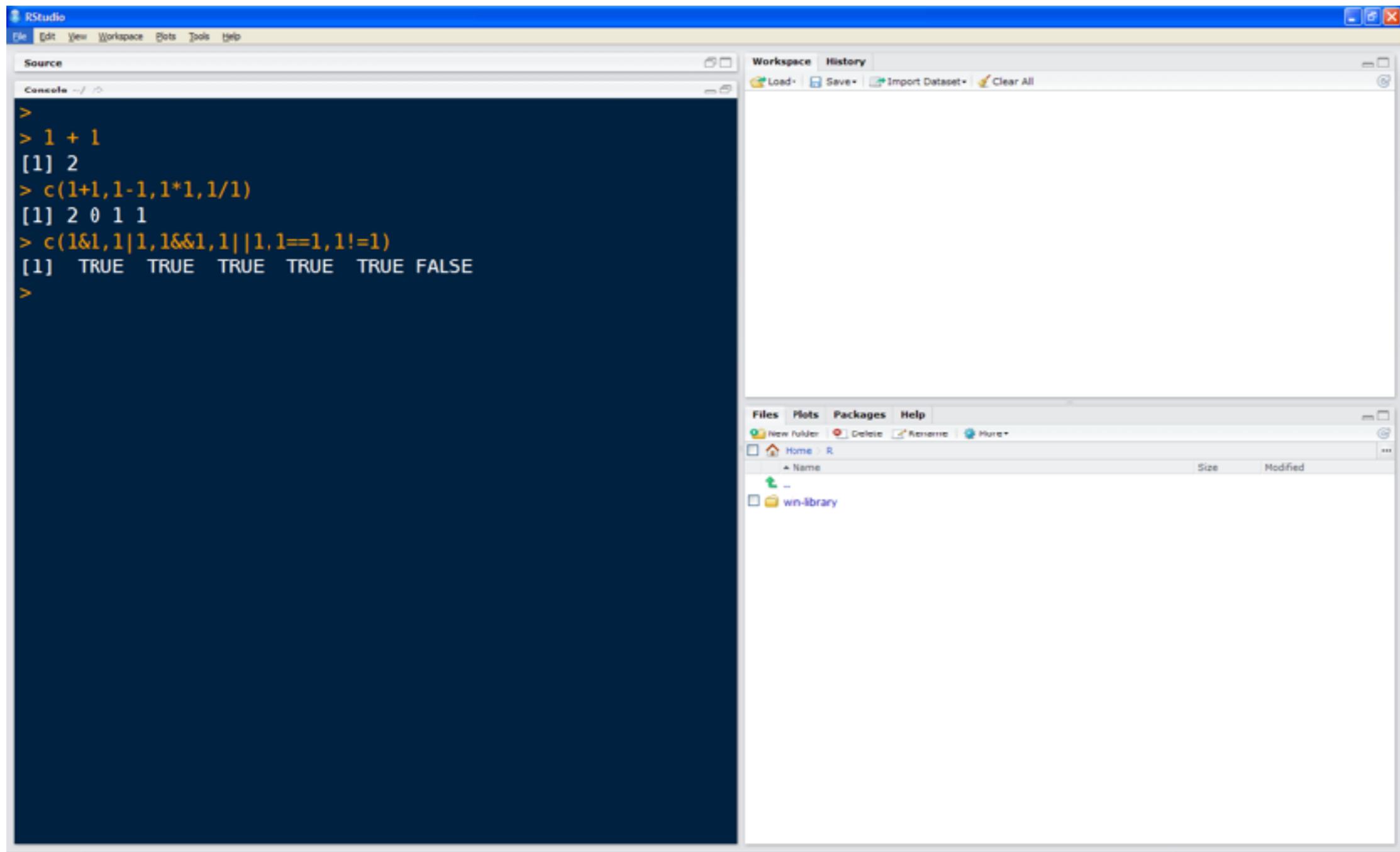
Working Directory



Ready to Code!

- The working directory is where Rstudio will look first for scripts
- Keeping everything in a self contained directory helps organize code and analyses
- Check you current working directory with
 - `getwd()`

R as a Calculator



Basic Math / Basic Logic

- + addition
- subtraction
- * multiplication
- / division
- % modulus (remainder)
- ^ to the power

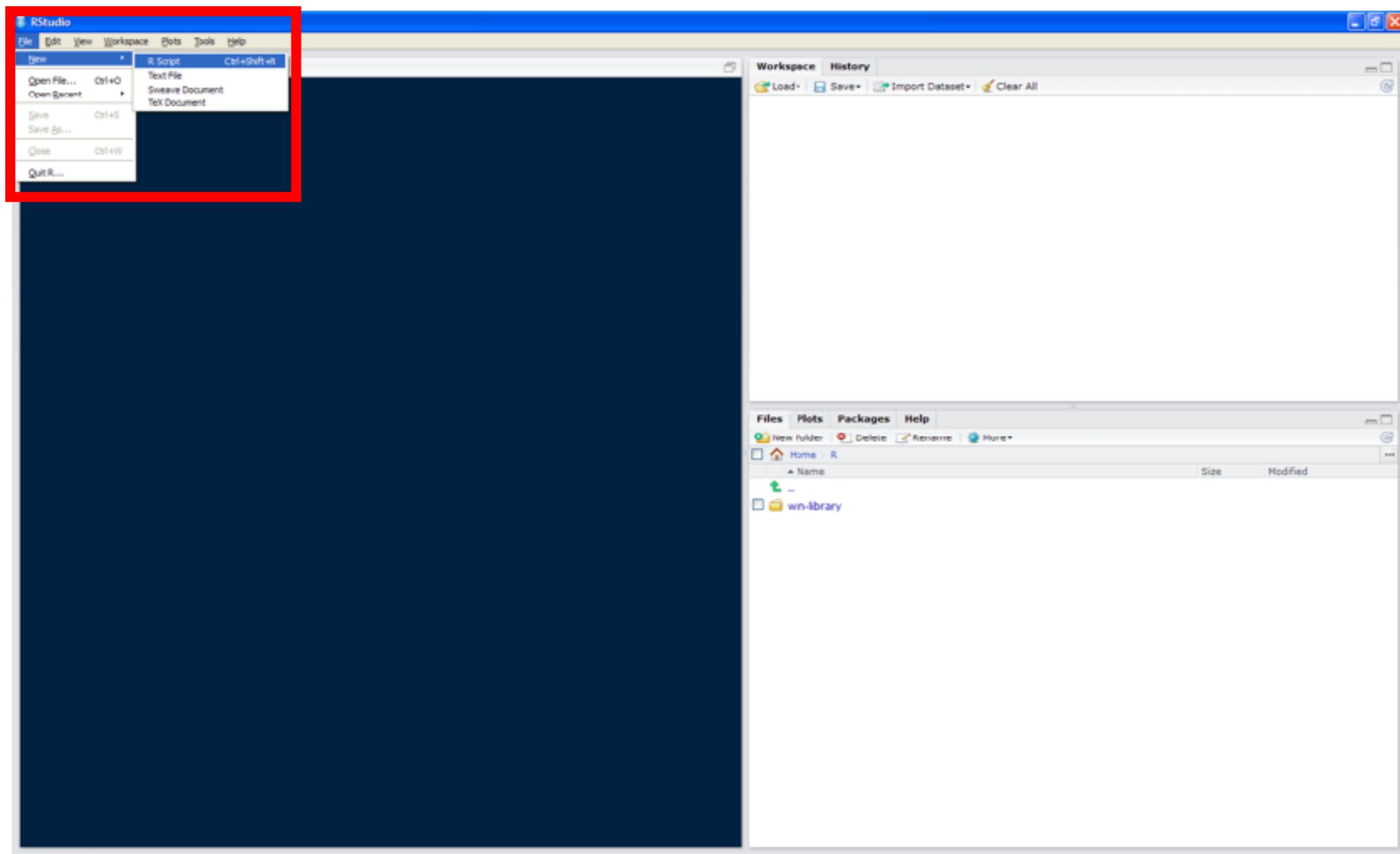
?Arithmetic

- ! NOT
- & bitwise AND
- | bitwise OR
- && shortcircuit AND
- || shortcircuit OR
- == equality
- != NOT equality

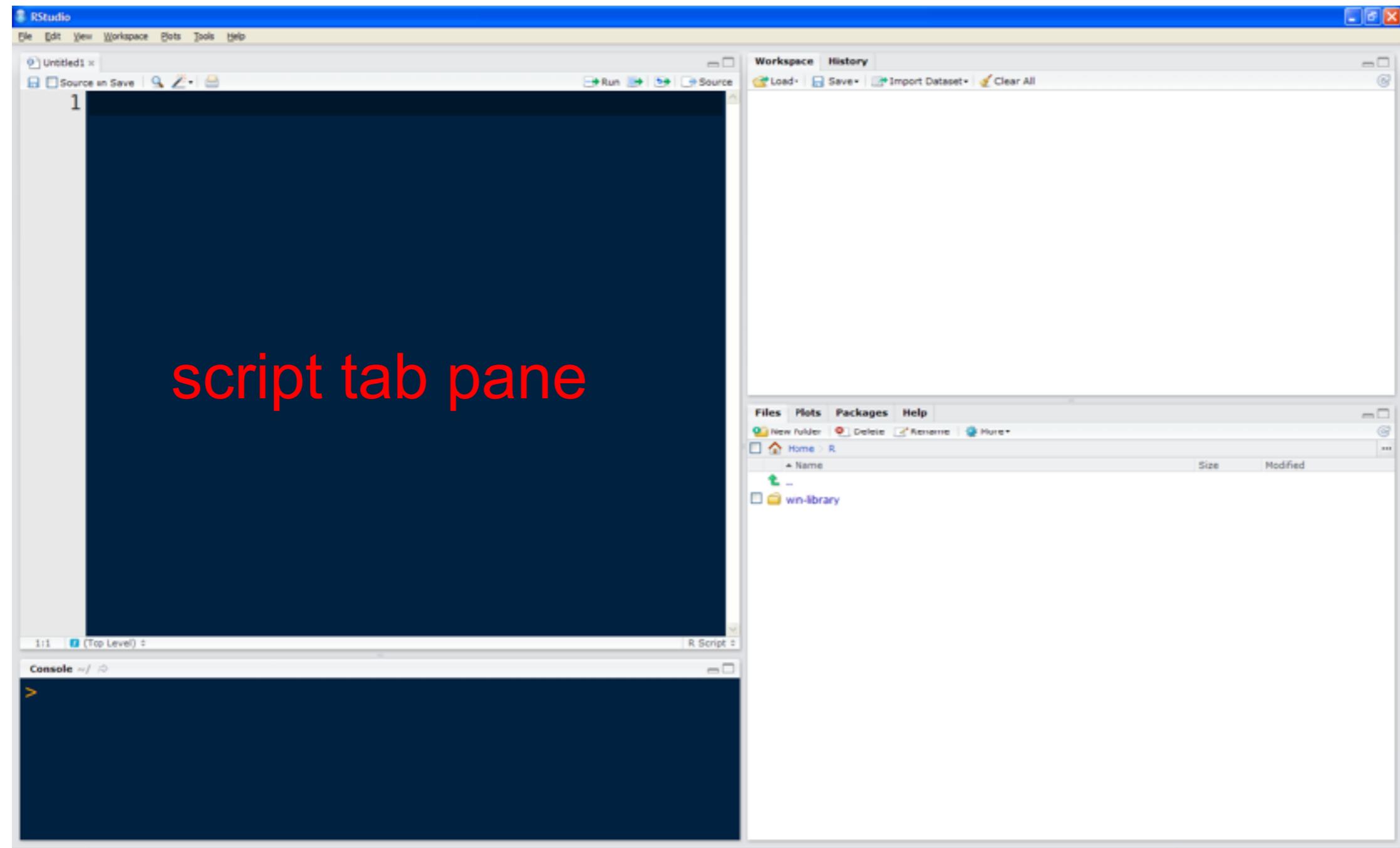
?Logic

Also try ?Syntax, ?Comparison

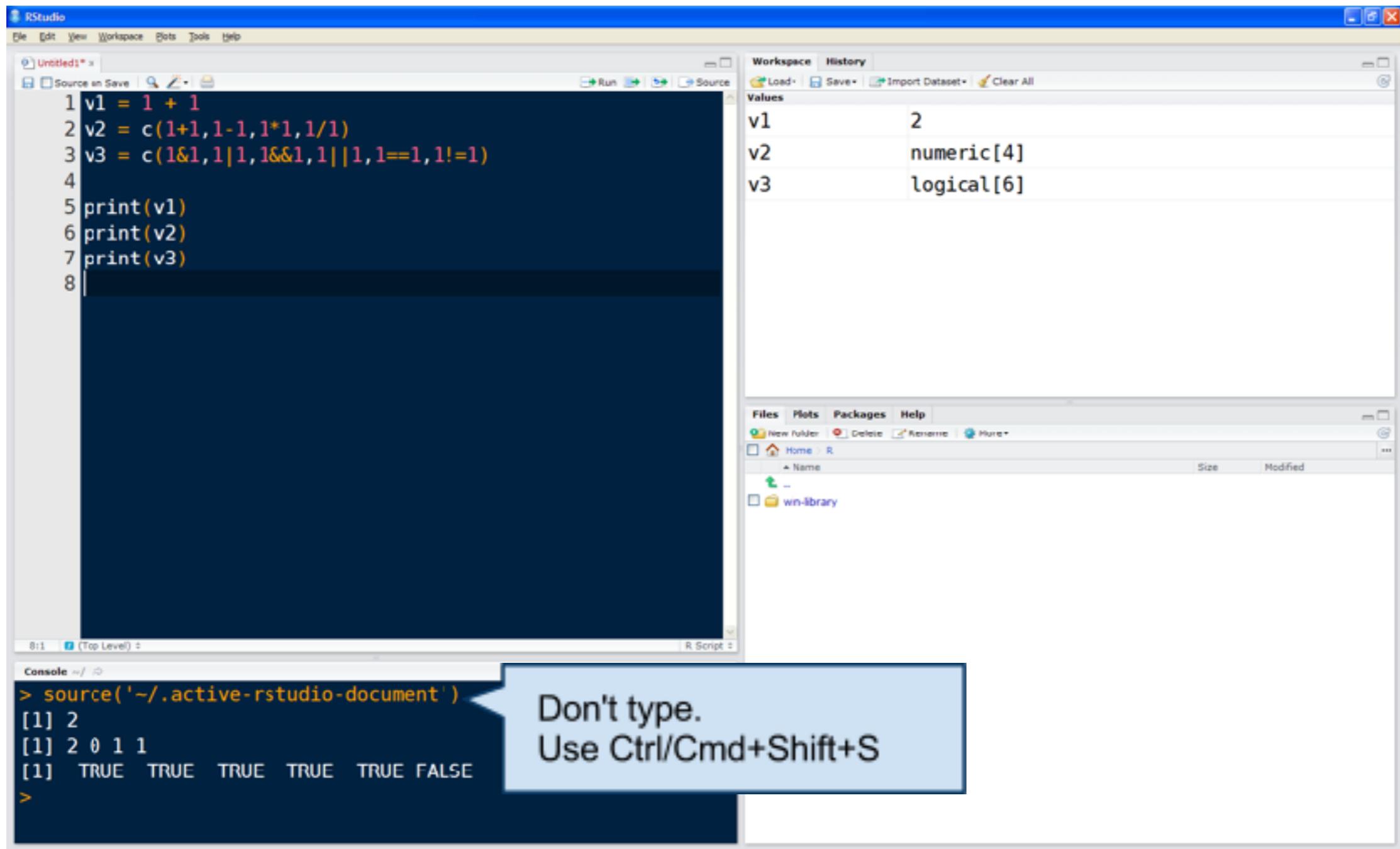
Make New R Script



Make New R Script



Run Script Loaded in Rstudio



```
1 v1 = 1 + 1
2 v2 = c(1+1,1-1,1*1,1/1)
3 v3 = c(1&1,1|1,1&&1,1||1,1==1,1!=1)
4
5 print(v1)
6 print(v2)
7 print(v3)
8
```

Values

- v1
- v2 numeric[4]
- v3 logical[6]

Save Workspace As...
Save As Default Workspace

```
> source('~/active-rstudio-document')
[1] 2
[1] 2 0 1 1
[1] TRUE TRUE TRUE TRUE TRUE FALSE
>
```

Saving Your Workspace

- In R you can save your entire workspace, variables and all in its current state
- Then you can reload this at a later time or can provide this to collaborators
- Workspace data files are saved with the extension ‘.Rdata’

Getting Help

- Inside of R
- Simplify make a fake dataset
- `help(<function name>)`
- `Help.start()`
- `?<function name>`
- `??<search term>`
- On the web
 - www.rseek.org and R only search engine
 - CRAN
 - Google topic with CRAN

Creating Variables

- Named containers for data
- Names can be anything you like except for 'special' words
- Better if names describe what the stored data is
Creating / Setting variables is a matter of equality:
 - **var = somedata**

Data Types in R

- R's atomic data type is the **vector**
 - numeric
 - floating point
 - integer
 - logical
 - character
 - **functions**
 - **lists**
 - let you combine other data types
- ```
> a <- 101
> length(a)
[1] 1
> a[1]
[1] 101
> a[2]
[1] NA
> a[2] <- 202
> length(a)
[1] 2
```

# Reading and Getting Data into R

## Combine Command

```
c(item.1, item.2, item.3, item.n)
```

```
c("item1", "item2", "item3")
```

## Scan Command

```
our.data = scan()
```

```
scan(what = 'character')
data5 = scan(sep = ',', what = 'char')
data6 = scan(file = 'test data.txt')
```

## Working Directory

```
getwd()
```

```
setwd('pathname')
```

# Reading Bigger Data Files

```
read.csv()
read.csv(file, sep = ',', header = TRUE, row.names)
fw = read.csv(file.choose())

my.ssv = read.table(file.choose(), header = TRUE)
my.tsv = read.delim(file.choose())
my.tsv = read.csv(file.choose(), sep = '\t')
my.tsv = read.table(file.choose(), header = TRUE,
sep = '\t')
```

# Convert between number and text data

```
cut2 = as.character(cut)
```

```
cut3 = as.factor(cut2)
```

```
data7i = as.integer(data7)
```

```
data7n = as.numeric(data7i)
```

# Selecting and displaying parts of a vector

| COMMAND                                         | RESULT                                               |
|-------------------------------------------------|------------------------------------------------------|
| <code>data1[1]</code>                           | Shows the first item in the vector.                  |
| <code>data1[3]</code>                           | Shows the third item.                                |
| <code>data1[1:3]</code>                         | Shows the first to the third items.                  |
| <code>data1[-1]</code>                          | Shows all except the first item.                     |
| <code>data1[c(1, 3, 4, 8)]</code>               | Shows the items listed in the <code>c()</code> part. |
| <code>data1[data1 &gt; 3]</code>                | Shows all items greater than 3.                      |
| <code>data1[data1 &lt; 5   data1 &gt; 7]</code> | Shows items less than 5 or greater than 7.           |

# Workshop : R Exercise 1

Assume that we have registered the height and weight for four people:

Heights in cm are 180, 165, 160, 193; weights in kg are 87, 58, 65, 100.

Make two vectors, height and weight, with the data. The bodymass index (BMI) is defined as

$$\text{weight in kg} / (\text{height in m})^2$$

Make a vector with the BMI values for the four people, and a vector with the natural logarithm to the BMI values. Finally make a vector with the weights for those people who have a BMI larger than 25.

# Coding Style

- Coding style (and code commenting) will become increasingly more important as we get into more advanced and involved programming tasks
- A few R “style guides” exist:
  - <http://r-pkgs.had.co.nz/style.html>
  - Google’s R Style Guide (<https://google.github.io/styleguide/Rguide.xml>)

# Data Frames

2 Dimensional Objects, it has rows and columns. R treats the columns as separate samples or variables, rows represent the replicates or observations.

```
> grass
 species cut
 1 12 mow
 2 15 mow
 3 17 mow
 4 11 mow
 5 15 mow
 6 8 unmow
 7 9 unmow
 8 7 unmow
 9 9 unmow
```

# Matrix Objects

A matrix is a two-dimensional data object. At first glance a matrix looks just like a data frame:

```
> bird
```

|               | Garden | Hedgerow | Parkland | Pasture | Woodland |
|---------------|--------|----------|----------|---------|----------|
| Blackbird     | 47     | 10       | 40       | 2       | 2        |
| Chaffinch     | 19     | 3        | 5        | 0       | 2        |
| Great Tit     | 50     | 0        | 10       | 7       | 0        |
| House Sparrow | 46     | 16       | 8        | 4       | 0        |
| Robin         | 9      | 3        | 0        | 0       | 2        |
| Song Thrush   | 4      | 0        | 6        | 0       | 0        |

# Structure of Objects

- `str()`
- To Examine the structure of an object
- `class()`
- Tell you the class of object

# Saving Data Files to Disk

`save()`

`save(list, file = ‘filename’)`

`save(bf, bf.lm, bf.beta, file = ‘desktop/butterly.rdata’)`

`save(list = ls(pattern = ‘^bf’), file = ‘desktop/butterfly.rdata’)`

`save(list = ls(all=TRUE), file=‘filename’)`

`save.image(file=‘filename’)`

# Reading Data Files from Disk

```
load(file='filename.Rdata')
load(file=file.choose())
```

# Save Data to disk as text files

```
write(x, file="data", sep=".")
```

Save Data Frame or Matrix

```
Write.table(mydata, file='filename', row.names=TRUE, sep=' ',
col.names=TRUE)
```

# Sorting and rearranging a vector

sort()

sort(unmow, decreasing =TRUE)

Get an Index using order()

order(unmow)

# Logical Values from a vector

```
data1 = c(3, 5, 7, 3, 2, 6)
which(data1 == 6)
```

Quicklooks

```
head(mf)
head(mf, n = 3)
Give simple Stat
summary(bird.m)
```

# Rotating Data Tables

```
fw.t = t(fw)
```

## Making Data Frames

```
My.frame = data.frame(item1, item2, item3)
```

## Making Matrix Objects

```
cmat = cbind(sampl1, sampl2)
```

# Vignettes

Some packages have vignettes

A vignette is an example of how to run the code and a lot of additional text explaining a lot more than you may want to know

List all available vignettes:

```
vignette()
```

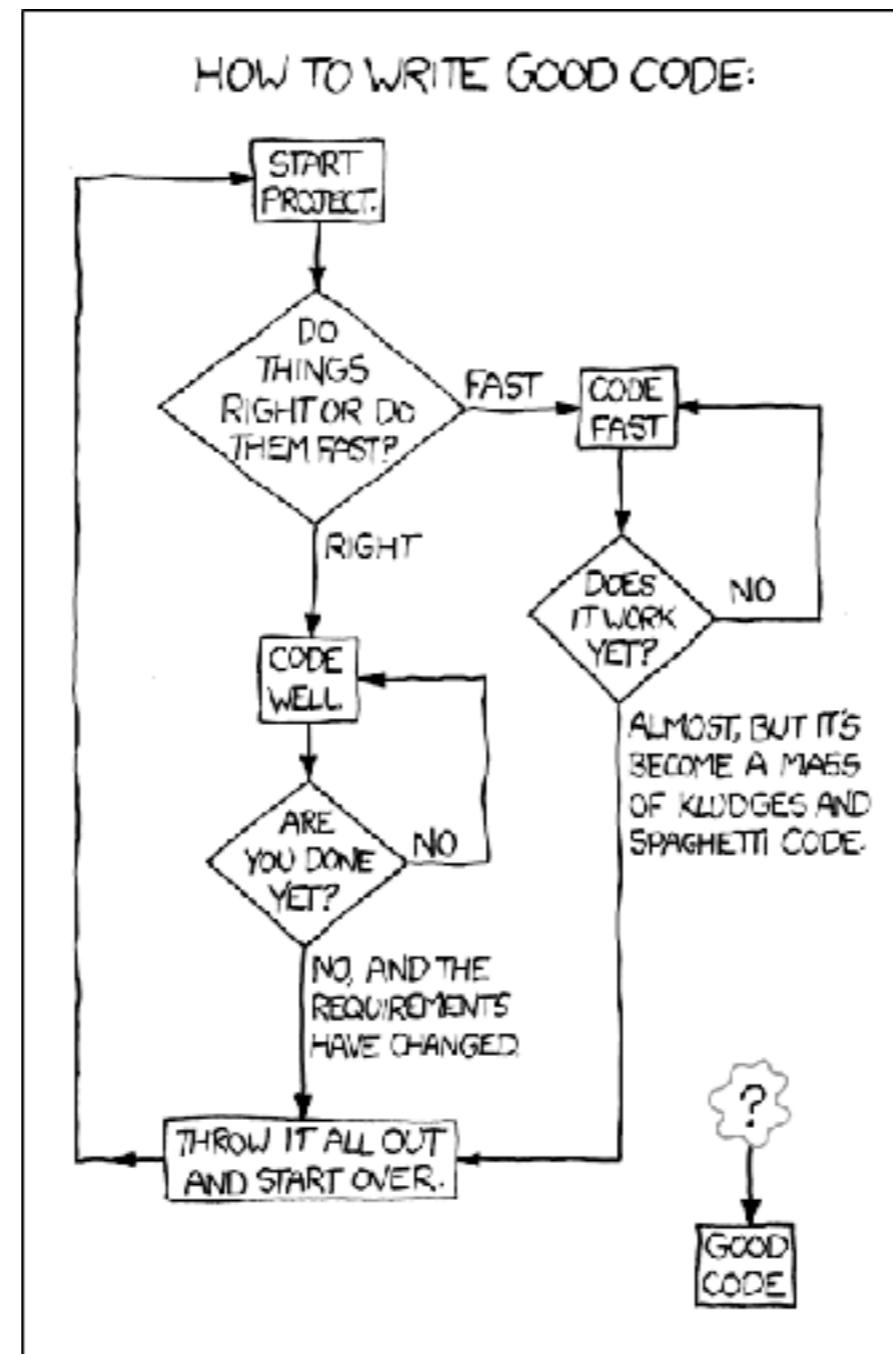
Display the vignette as a pdf by executing

```
vignette('<topic>')
```

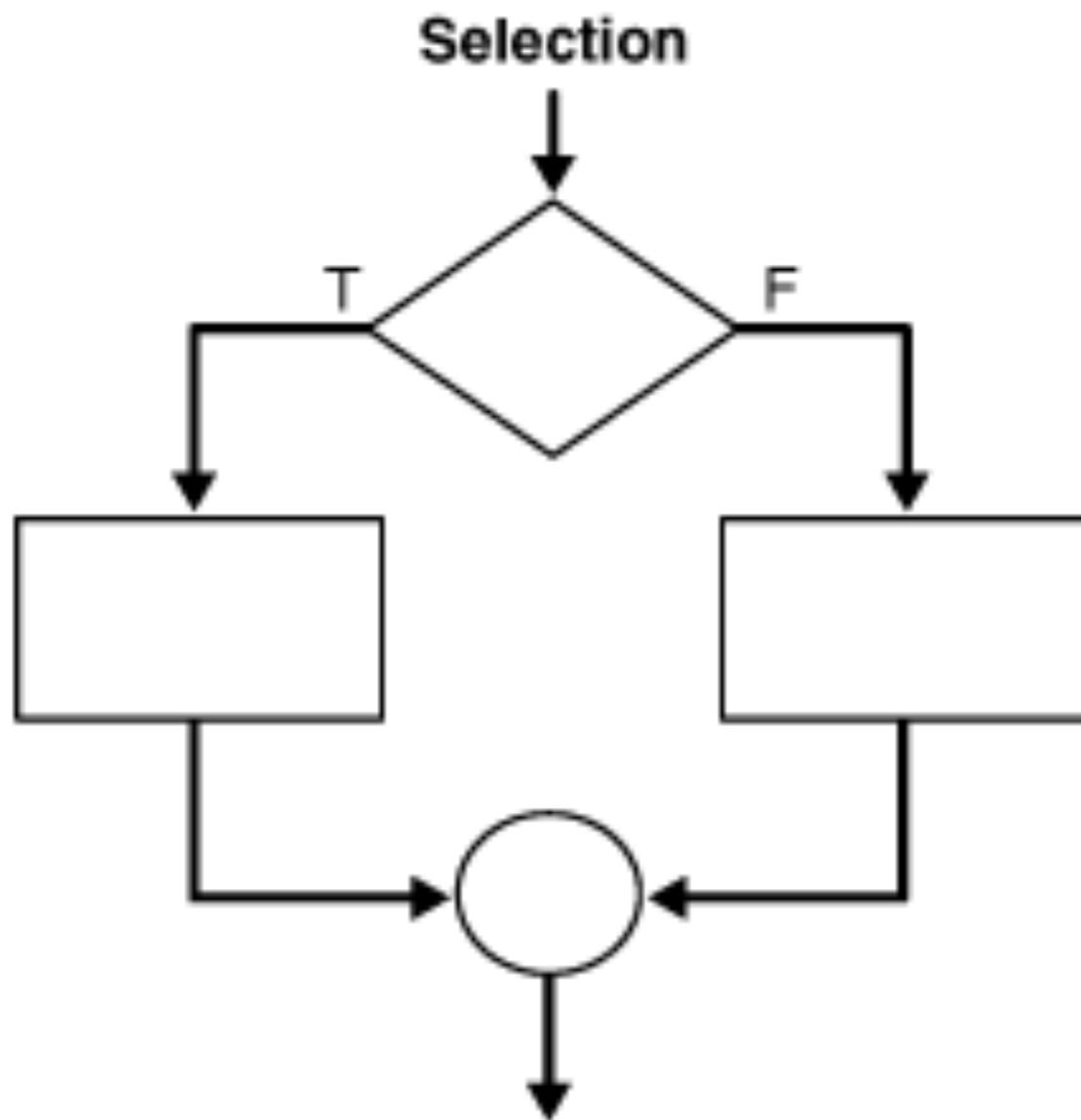
To play with the vignette code

```
vig = vignette('<topic>')
edit(vig)
```

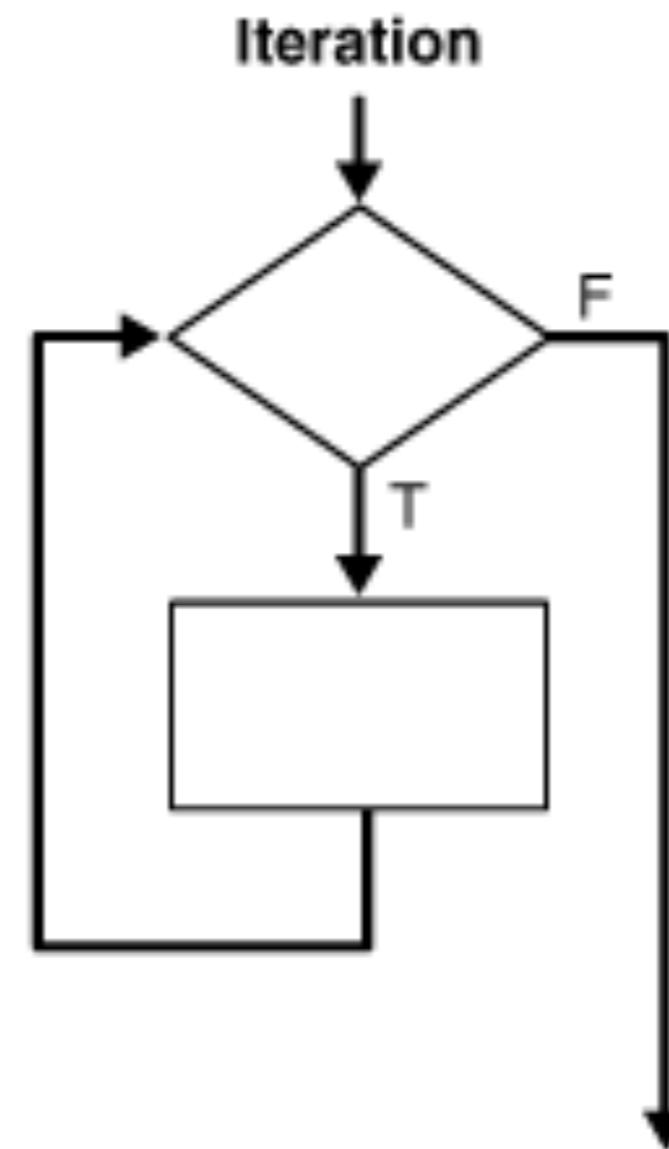
# Coding



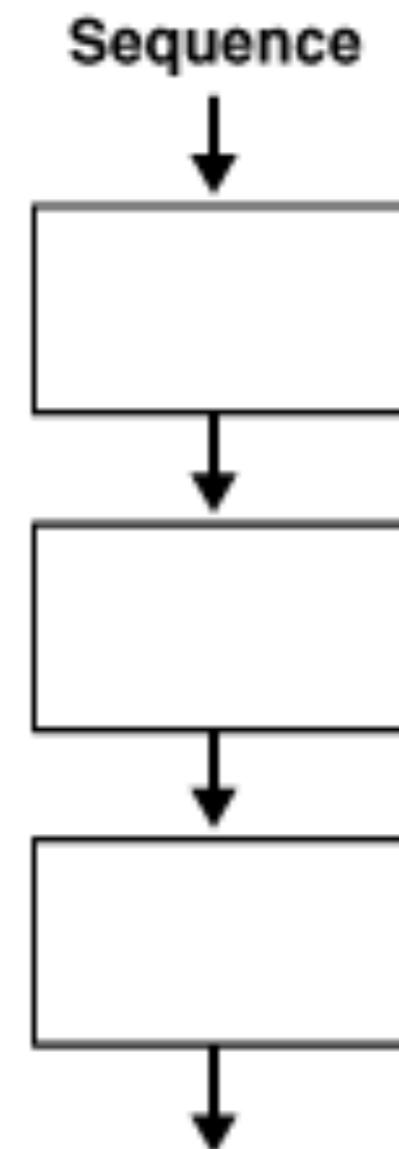
# Programmatic Structure



if ... then ... else



for ...  
while ...



# For Loop

Repeats a line or lines (known as a block) of code until:  
(for loop) a count limit is reached

```
for (i in 1:10) {
 ... code ...
}
```

the above loop runs 10 times

{' and '}' enclose code looped

the variable i updated in the loop to values in the sequence  
1:10

# A Simple For Loop

```
P-values from our analysis
p.values = c(0.1, 0.05, 0.003, 0.4, 0.9)

A vector to store the negative log p-values
neglog10.p.values = 1:5

Transform the p-values
for(p in 1:length(p.values)) {
 neglog10.p.values[p] = -log10(p.values[p])
}
neglog10.p.values.2 = -log10(p.values)
```

# While Loop

Repeats a line or lines (known as a block) of code until:  
(while loop) a logical condition is reached

```
while (stop != TRUE) {
 ... code ...
}
```

the above loop runs until code sets  
stop = TRUE

**warning:** if not properly written while loops can run  
infinitely

# A Simple While Loop

```
Numbers from our analysis
v1 = c(21, 22, 53, 74, 85, 96, 97, 58, 49, 30, 85)

Iterator
i = 1

Look for first instance of 85
while(v1[i] != 85) {
 i = i + 1
}

Print out where we found it
print(paste('v1[', i, '] = 85', sep=''))
```

# Functions

Bits of code that do one thing and (preferably) do it well

Functions break up your code into more manageable and reusable parts

Defining (e.g. in a script):

```
fun = function(arguments) {
 ... code ...
}
```

Calling:

```
party = fun(food,beer,folks)
```

# A Simple Function

```
fn1 <- function(N) {
 for(i in as.numeric(1:N)) {
 y <- i*i
 }
}

fn2 <- function(N) {
 i = 1
 while(i <= N) {
 y <- i*i
 i <- i + 1
 }
}

system.time(fn1(60000))
system.time(fn2(60000))
```

# Workshop : R Exercise 2

You probably most often save your data in Excel (or something similar), and therefore need to “transfer” your data from Excel to R. There is safe way and a quick, but not so safe, way. “Not so safe” means that problems quite often occur due to different versions of Excel and different computer systems.

The safe way to do so consists of two steps:

- Save your Excel sheet as a comma separated file (.csv)
- Read the csv file into a R dataset with `read.csv`

# Workshop : R Exercise 2

Consider an experiment where girth, height and volume has been measured for 31 cherry trees. The data are saved in the file `cherry.xlsx`.

1. Open the file `cherry.xlsx` in Excel (or a similar application), and make sure you understand the structure of the file.
2. Go to the file menu, and save the data as a csv file (use Save As). As a starting point, you can use the default options for delimiters.
3. Now, go to R and use the command

```
cherry <- read.csv(file.choose(), dec=',', sep=';')
```

and choose the relevant csv file. You may write the file name (including the path to the file) instead of `file.choose()`, i.e. '`cherry.csv`'. Notice how you can change the characters for decimal comma and field separator according to what is used in the csv file, so the file is interpreted in the correct way.



# Visualization in R

Fundamental Data Science for Data Scientist

# Data for Practice

<https://github.com/vkrit/data-science-class>

<http://google-styleguide.googlecode.com/svn/trunk/Rguide.xml>

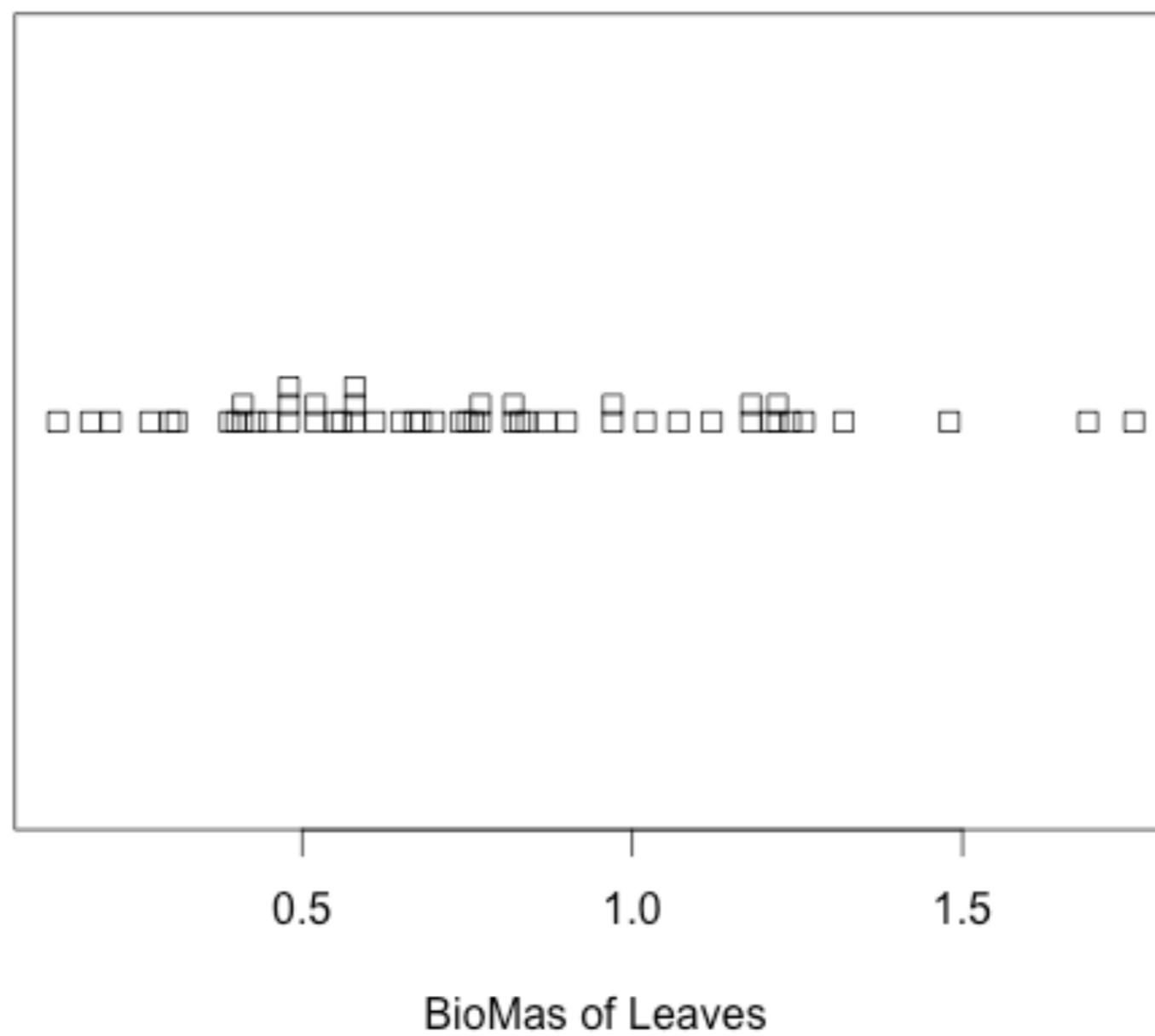
# Basic Plots

```
> w1 <- read.csv(file="w1.dat",sep=",",head=TRUE)
> names(w1)
[1] "vals"
> tree <- read.csv(file="trees91.csv",sep=",",head=TRUE)
> names(tree)
[1] "C" "N" "CHBR" "REP" "LFBM" "STBM" "RTBM" "LFNCC"
[9] "STNCC" "RTNCC" "LFBCC" "STBCC" "RTBCC" "LFCACC" "STCACC" "RTCACC"
[17] "LFKCC" "STKCC" "RTKCC" "LFMGCC" "STMGCC" "RTMGCC" "LFPCC" "STPCC"
[25] "RTPCC" "LFSCC" "STS CC" "RTSCC"
```

```
> stripchart(w1$vals,vertical=TRUE)
> stripchart(w1$vals,vertical=TRUE,method="jitter")
```

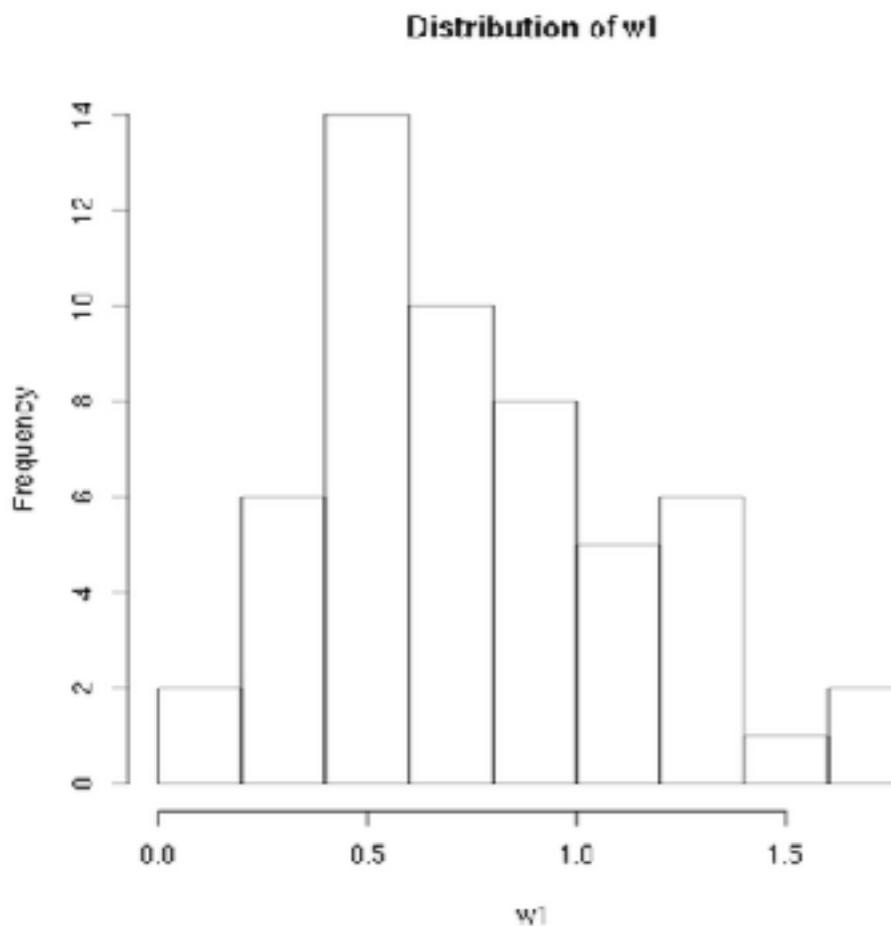
```
> stripchart(w1$vals,method="stack",
 main='Leaf BioMass in High CO2 Environment',
 xlab='BioMass of Leaves')
```

## Leaf BioMass in High CO<sub>2</sub> Environment



# Histogram

```
> hist(w1$vals)
> hist(w1$vals,main="Distribution of w1",xlab="w1")
```



```
> hist(w1$vals,
 main='Leaf BioMass in High CO2 Environment',
 xlab='BioMass of Leaves')
```

```
> title('Leaf BioMass in High CO2 Environment',xlab='BioMass of Leaves')
```

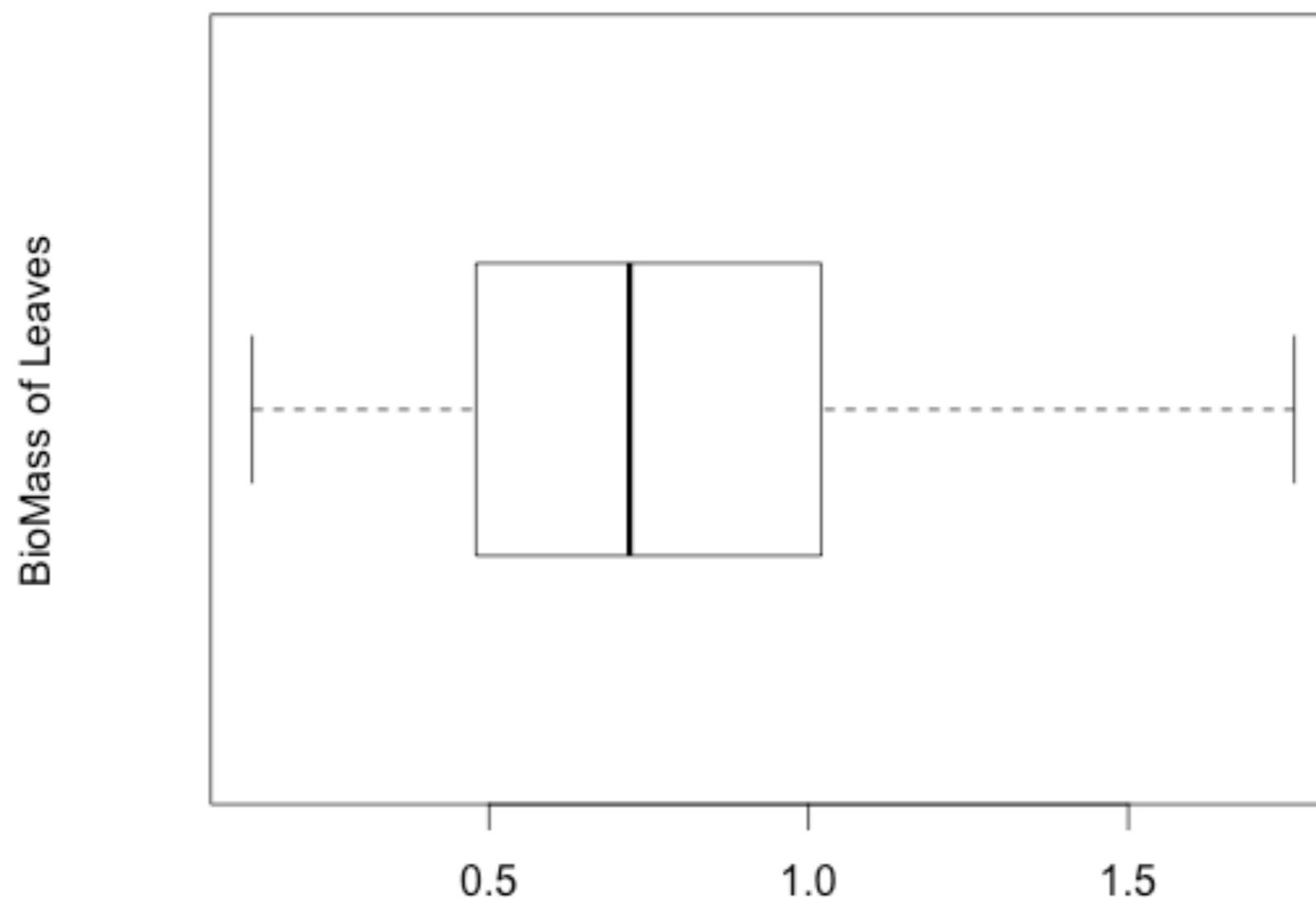
# Boxplot

```
> boxplot(w1$vals)

> boxplot(w1$vals,
 main='Leaf BioMass in High CO2 Environment',
 ylab='BioMass of Leaves')

> boxplot(w1$vals,
 main='Leaf BioMass in High CO2 Environment',
 xlab='BioMass of Leaves',
 horizontal=TRUE)
```

## Leaf BioMass in High CO<sub>2</sub> Environment

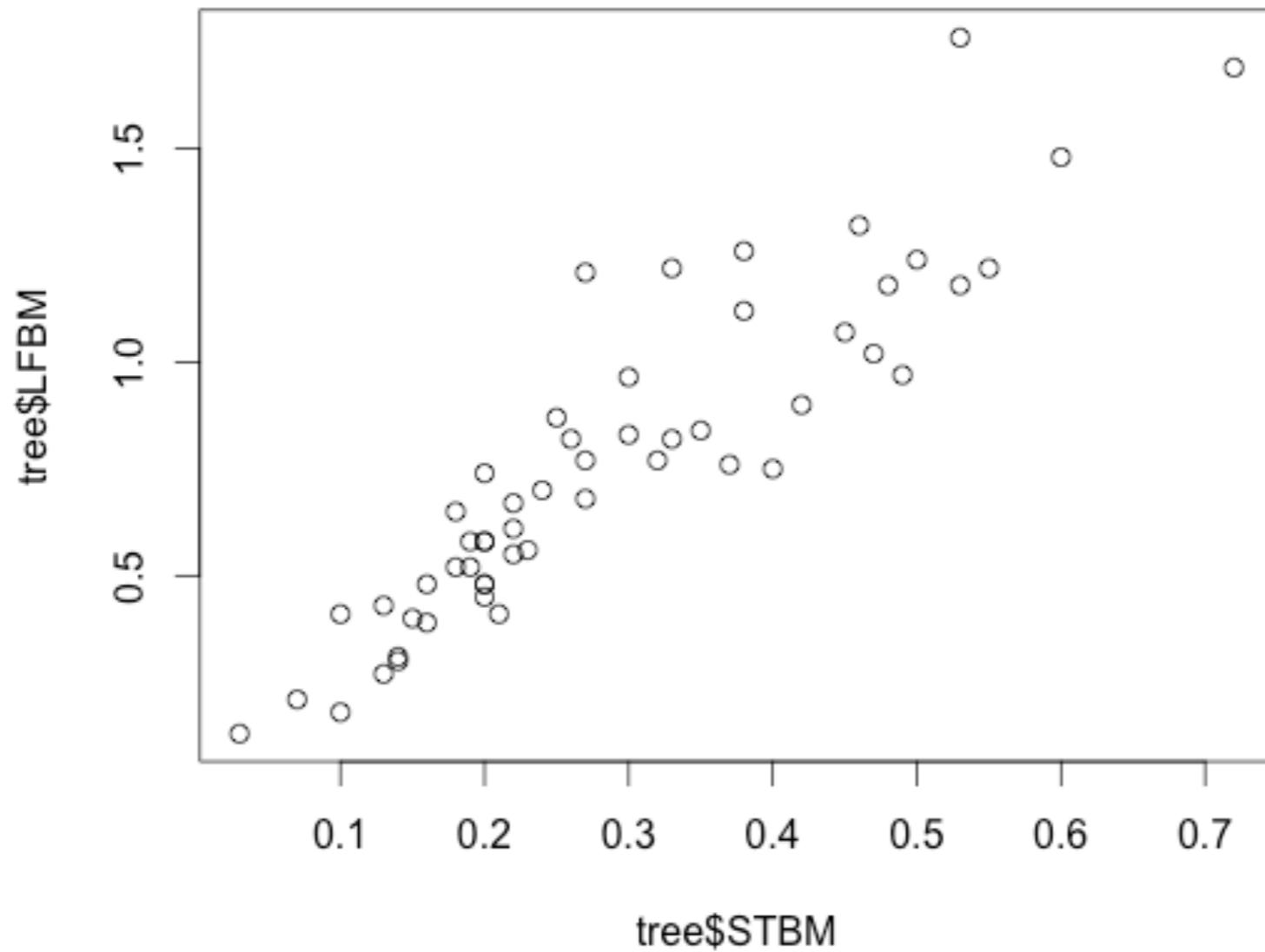


# Scatter Plot

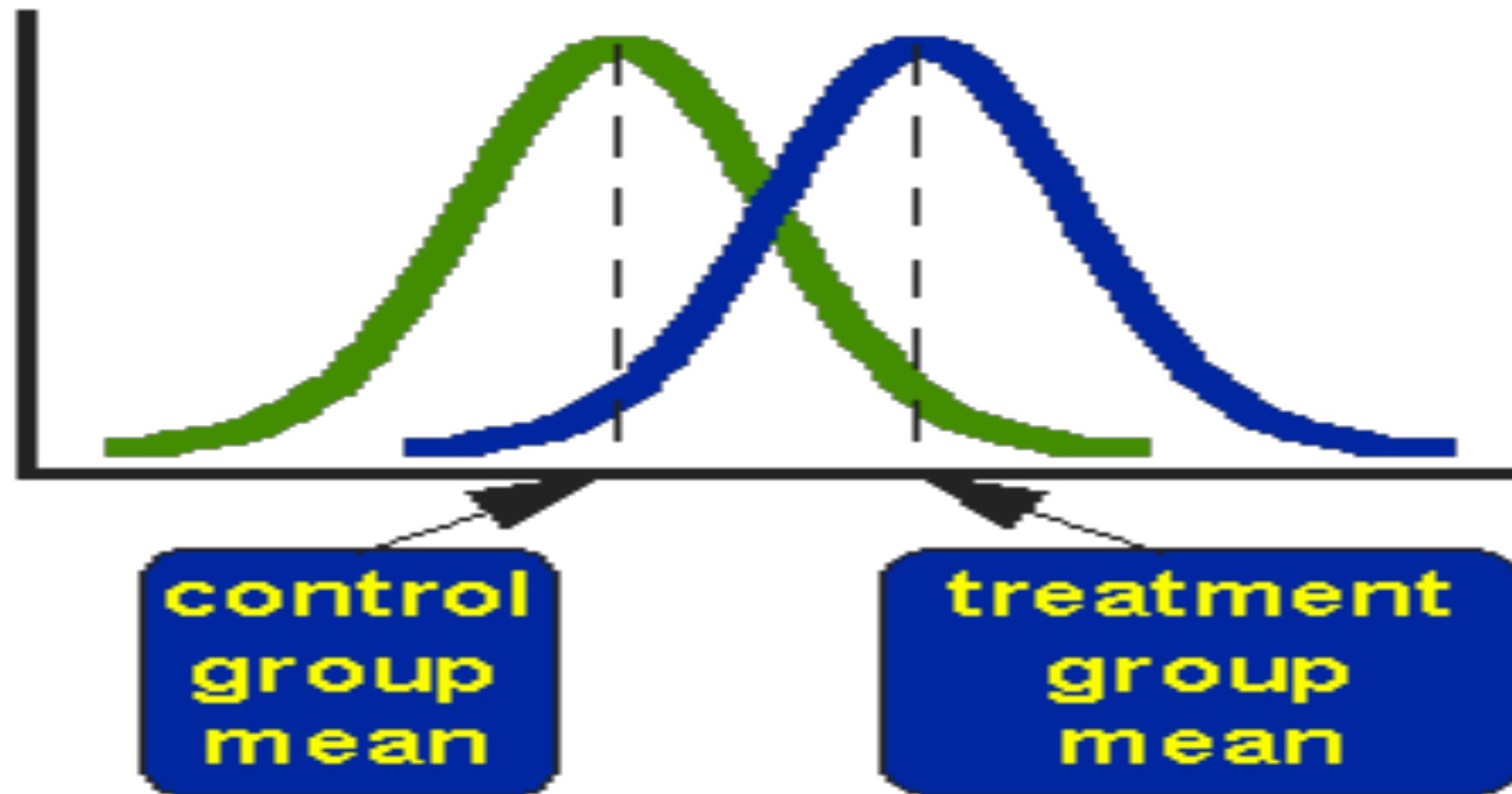
```
> plot(tree$STBM,tree$LFBM)
```

```
> cor(tree$STBM,tree$LFBM)
[1] 0.911595
```

```
> plot(tree$STBM,tree$LFBM,
 main="Relationship Between Stem and Leaf Biomass",
 xlab="Stem Biomass",
 ylab="Leaf Biomass")
```



# Student's T-test



# Comparing the Means of Two Groups

Sleep dataset in R is a comparison of two soporific drugs on students sleep habits

```
> sleep
 extra group ID
1 0.7 1 1
2 -1.6 1 2
3 -0.2 1 3
4 -1.2 1 4
5 -0.1 1 5
6 3.4 1 6
7 3.7 1 7
8 0.8 1 8
9 0.0 1 9
10 2.0 1 10
11 1.9 2 1
12 0.8 2 2
13 1.1 2 3
14 0.1 2 4
15 -0.1 2 5
16 4.4 2 6
17 5.5 2 7
18 1.6 2 8
19 4.6 2 9
20 3.4 2 10
> |
```

# Doing a Student's T-test

```
t.test(sleep[1:10,'extra'],sleep[11:20,'extra'])
```

or

```
t.test(extra ~ group, data = sleep)
```

```
> t.test(extra ~ group, data = sleep)
```

```
welch Two Sample t-test
```

```
data: extra by group
```

```
t = -1.8608, df = 17.776, p-value = 0.07939
```

```
alternative hypothesis: true difference in means is not equal to 0
```

```
95 percent confidence interval:
```

```
-3.3654832 0.2054832
```

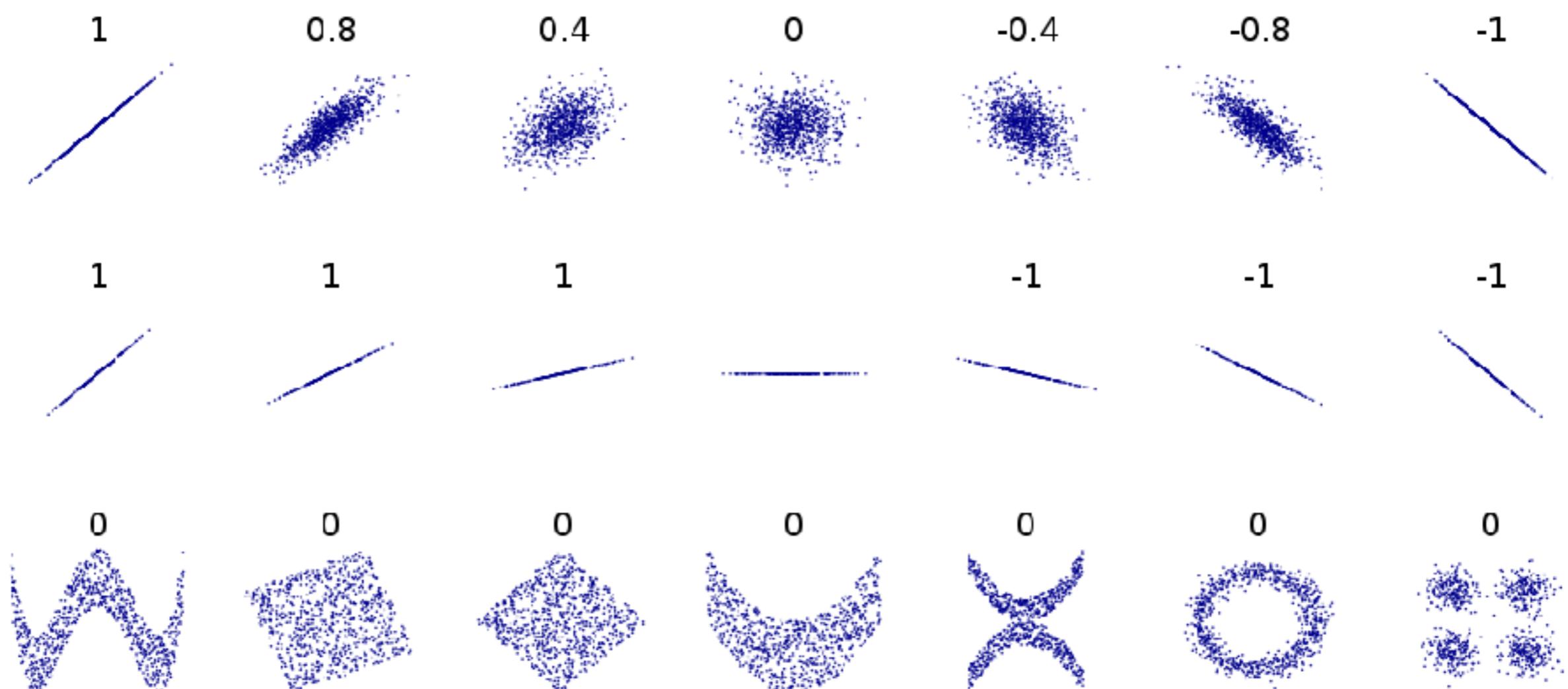
```
sample estimates:
```

```
mean in group 1 mean in group 2
0.75 2.33
```

```
>
> |
```

| > sleep |       |       |    |
|---------|-------|-------|----|
|         | extra | group | ID |
| 1       | 0.7   | 1     | 1  |
| 2       | -1.6  | 1     | 2  |
| 3       | -0.2  | 1     | 3  |
| 4       | -1.2  | 1     | 4  |
| 5       | -0.1  | 1     | 5  |
| 6       | 3.4   | 1     | 6  |
| 7       | 3.7   | 1     | 7  |
| 8       | 0.8   | 1     | 8  |
| 9       | 0.0   | 1     | 9  |
| 10      | 2.0   | 1     | 10 |
| 11      | 1.9   | 2     | 1  |
| 12      | 0.8   | 2     | 2  |
| 13      | 1.1   | 2     | 3  |
| 14      | 0.1   | 2     | 4  |
| 15      | -0.1  | 2     | 5  |
| 16      | 4.4   | 2     | 6  |
| 17      | 5.5   | 2     | 7  |
| 18      | 1.6   | 2     | 8  |
| 19      | 4.6   | 2     | 9  |
| 20      | 3.4   | 2     | 10 |

# Correlation



# Testing for a Correlation

```
cor.test(c(1:10), c(11:20))
```

```
> cor.test(c(1:10), c(11:20))
Pearson's product-moment correlation

data: c(1:10) and c(11:20)
t = 134217728, df = 8, p-value < 2.2e-16
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 1 1
sample estimates:
cor
 1
> |
```

# Testing for a Correlation

```
cor.test(c(1:10), -c(11:20))
```

```
> cor.test(c(1:10), -c(11:20))

Pearson's product-moment correlation

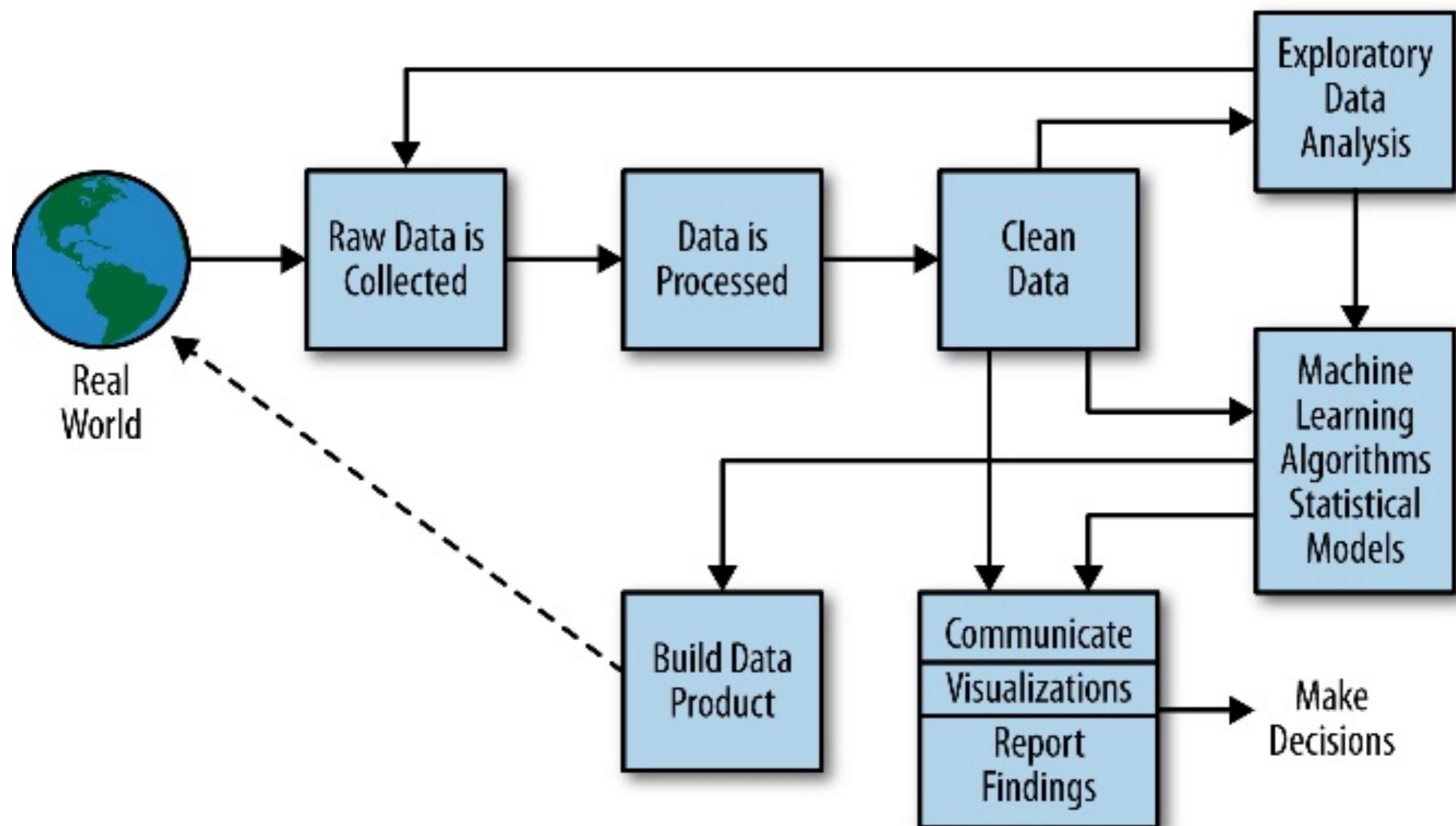
data: c(1:10) and -c(11:20)
t = -134217728, df = 8, p-value < 2.2e-16
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
-1 -1
sample estimates:
cor
-1

>
```

# What else is built-in?

```
library(help="stats")
```

# Data Science Life Cycle





# Machine Learning Techniques

Fundamental Data Science for Data Scientist

# Topic

- **Basic concepts**
- Decision tree induction
- Evaluation of classifiers
- Classification using association rules
- Naïve Bayesian classification
- Naïve Bayes for text classification
- K-nearest neighbor
- Ensemble methods: Bagging and Boosting
- Summary

# An example application

An emergency room in a hospital measures 17 variables (e.g., blood pressure, age, etc) of newly admitted patients.

**A decision is needed:** whether to put a new patient in an intensive-care unit.

Due to the high cost of ICU, those patients who may survive less than a month are given higher priority.

**Problem:** to predict **high-risk patients** and discriminate them from **low-risk patients**.

# Another application

A credit card company receives thousands of applications for new cards. Each application contains information about an applicant,

age

Marital status

annual salary

outstanding debts

credit rating

etc.

**Problem:** to decide whether an application should approved, or to classify applications into two categories, **approved** and **not approved**.

# Machine learning and our focus

- Like human learning from past experiences.
- A computer does not have “experiences”.
- A computer system learns from data, which represent some “past experiences” of an application domain.
- Our focus: learn a target function that can be used to predict the values of a discrete class attribute, e.g., approve or not-approved, and high-risk or low risk.
- The task is commonly called: Supervised learning, classification, or inductive learning.

# The data and the goal

**Data:** A set of data records (also called examples, instances or cases) described by

*k* attributes:  $A_1, A_2, \dots A_k$ .

a class: Each example is labelled with a pre-defined class.

**Goal:** To learn a classification model from the data that can be used to predict the classes of new (future, or test) cases/instances.

# An example: data (loan application)

Approved or not

| ID | Age    | Has_Job | Own_House | Credit_Rating | Class |
|----|--------|---------|-----------|---------------|-------|
| 1  | young  | false   | false     | fair          | No    |
| 2  | young  | false   | false     | good          | No    |
| 3  | young  | true    | false     | good          | Yes   |
| 4  | young  | true    | true      | fair          | Yes   |
| 5  | young  | false   | false     | fair          | No    |
| 6  | middle | false   | false     | fair          | No    |
| 7  | middle | false   | false     | good          | No    |
| 8  | middle | true    | true      | good          | Yes   |
| 9  | middle | false   | true      | excellent     | Yes   |
| 10 | middle | false   | true      | excellent     | Yes   |
| 11 | old    | false   | true      | excellent     | Yes   |
| 12 | old    | false   | true      | good          | Yes   |
| 13 | old    | true    | false     | good          | Yes   |
| 14 | old    | true    | false     | excellent     | Yes   |
| 15 | old    | false   | false     | fair          | No    |

# An example: the learning task

- Learn a classification model from the data
- Use the model to classify future loan applications into
  - Yes (approved) and
  - No (not approved)
- What is the class for following case/instance?

| Age   | Has_Job | Own_house | Credit-Rating | Class |
|-------|---------|-----------|---------------|-------|
| young | false   | false     | good          | ?     |

# Supervised vs. unsupervised

**Supervised learning: classification is seen as supervised learning from examples.**

- **Supervision:** The data (observations, measurements, etc.) are labeled with pre-defined classes. It is like that a “teacher” gives the classes (**supervision**).
- Test data are classified into these classes too.

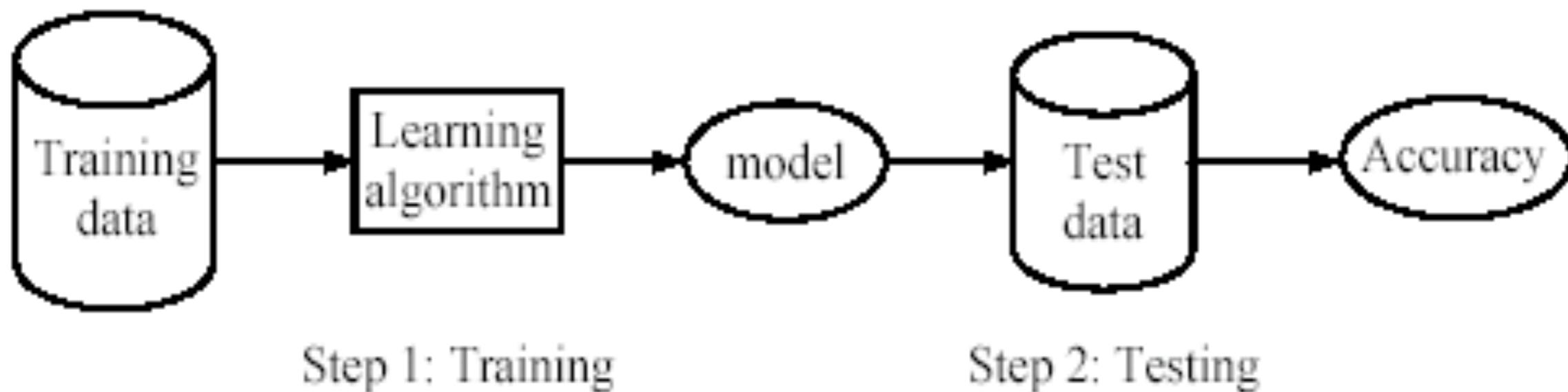
**Unsupervised learning (clustering)**

- **Class labels of the data are unknown**
- Given a set of data, the task is to establish the existence of classes or clusters in the data

# Supervised learning process: two

- **Learning (training):** Learn a model using the training data
- **Testing:** Test the model using **unseen test data** to assess the model accuracy

$$Accuracy = \frac{\text{Number of correct classifications}}{\text{Total number of test cases}},$$



# What do we mean by learning?

Given

a data set  $D$ ,

a task  $T$ , and

a performance measure  $M$ ,

a computer system is said to **learn** from  $D$  to perform the task  $T$  if after learning the system's performance on  $T$  improves as measured by  $M$ .

In other words, the learned model helps the system to perform  $T$  better as compared to no learning.

# An example

**Data:** Loan application data

**Task:** Predict whether a loan should be approved or not.

**Performance measure:** accuracy.

**No learning:** classify all future applications (test data) to the majority class (i.e., **Yes**):

$$\text{Accuracy} = 9/15 = 60\%.$$

We can do better than 60% with learning.

# Fundamental assumption of learning

**Assumption:** The distribution of training examples is identical to the distribution of test examples (including future unseen examples).

- In practice, this assumption is often violated to certain degree.
- Strong violations will clearly result in poor classification accuracy.
- To achieve good accuracy on the test data, training examples must be sufficiently representative of the test data.

# Topic

- Basic concepts
- **Decision tree induction**
- Evaluation of classifiers
- Classification using association rules
- Naïve Bayesian classification
- Naïve Bayes for text classification
- K-nearest neighbor
- Ensemble methods: Bagging and Boosting
- Summary

# Introduction

- Decision tree learning is one of the most widely used techniques for classification.
  - Its classification accuracy is competitive with other methods, and
  - it is very efficient.
- The classification model is a tree, called **decision tree**.
- **C4.5** by Ross Quinlan is perhaps the best known system. It can be downloaded from the Web.

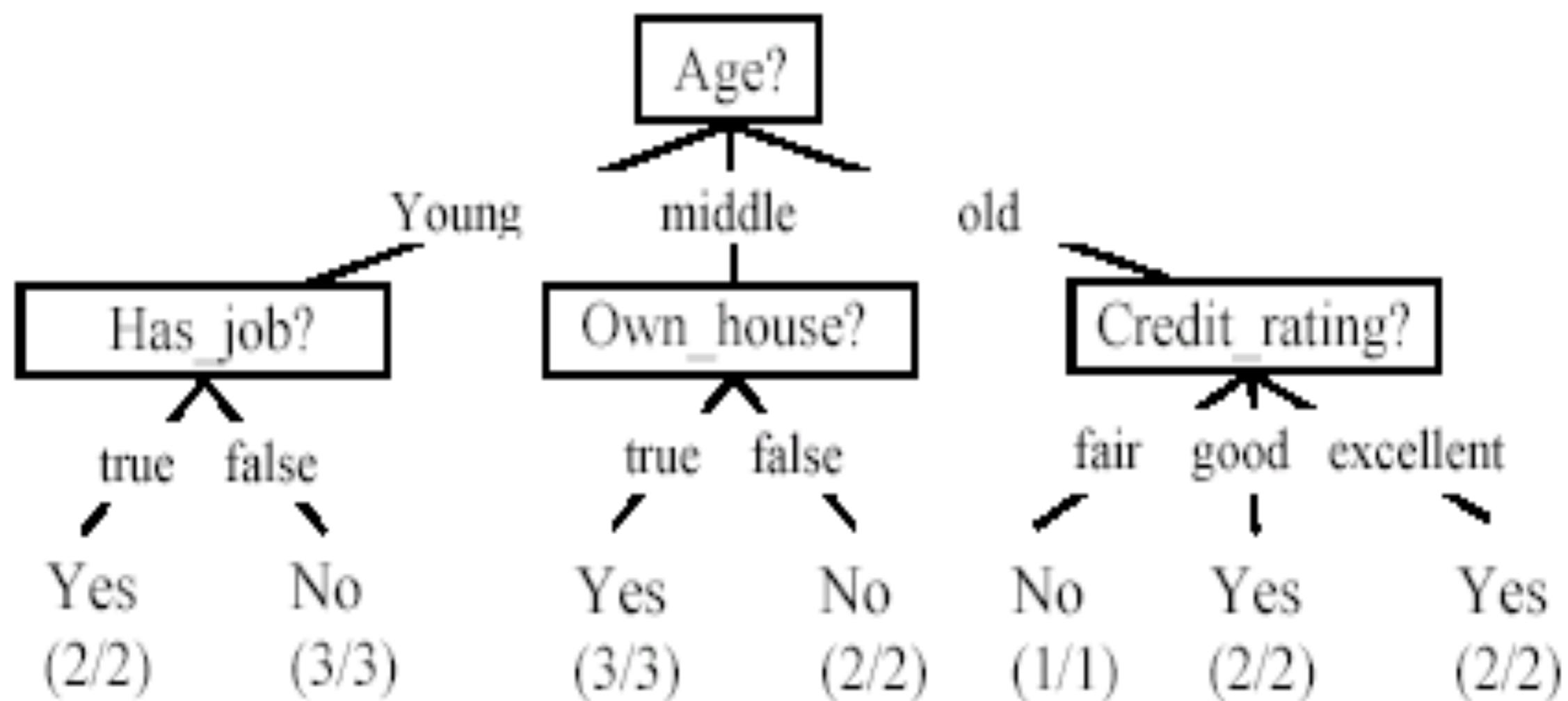
# The loan data (reproduced)

Approved or not

| ID | Age    | Has_Job | Own_House | Credit_Rating | Class |
|----|--------|---------|-----------|---------------|-------|
| 1  | young  | false   | false     | fair          | No    |
| 2  | young  | false   | false     | good          | No    |
| 3  | young  | true    | false     | good          | Yes   |
| 4  | young  | true    | true      | fair          | Yes   |
| 5  | young  | false   | false     | fair          | No    |
| 6  | middle | false   | false     | fair          | No    |
| 7  | middle | false   | false     | good          | No    |
| 8  | middle | true    | true      | good          | Yes   |
| 9  | middle | false   | true      | excellent     | Yes   |
| 10 | middle | false   | true      | excellent     | Yes   |
| 11 | old    | false   | true      | excellent     | Yes   |
| 12 | old    | false   | true      | good          | Yes   |
| 13 | old    | true    | false     | good          | Yes   |
| 14 | old    | true    | false     | excellent     | Yes   |
| 15 | old    | false   | false     | fair          | No    |

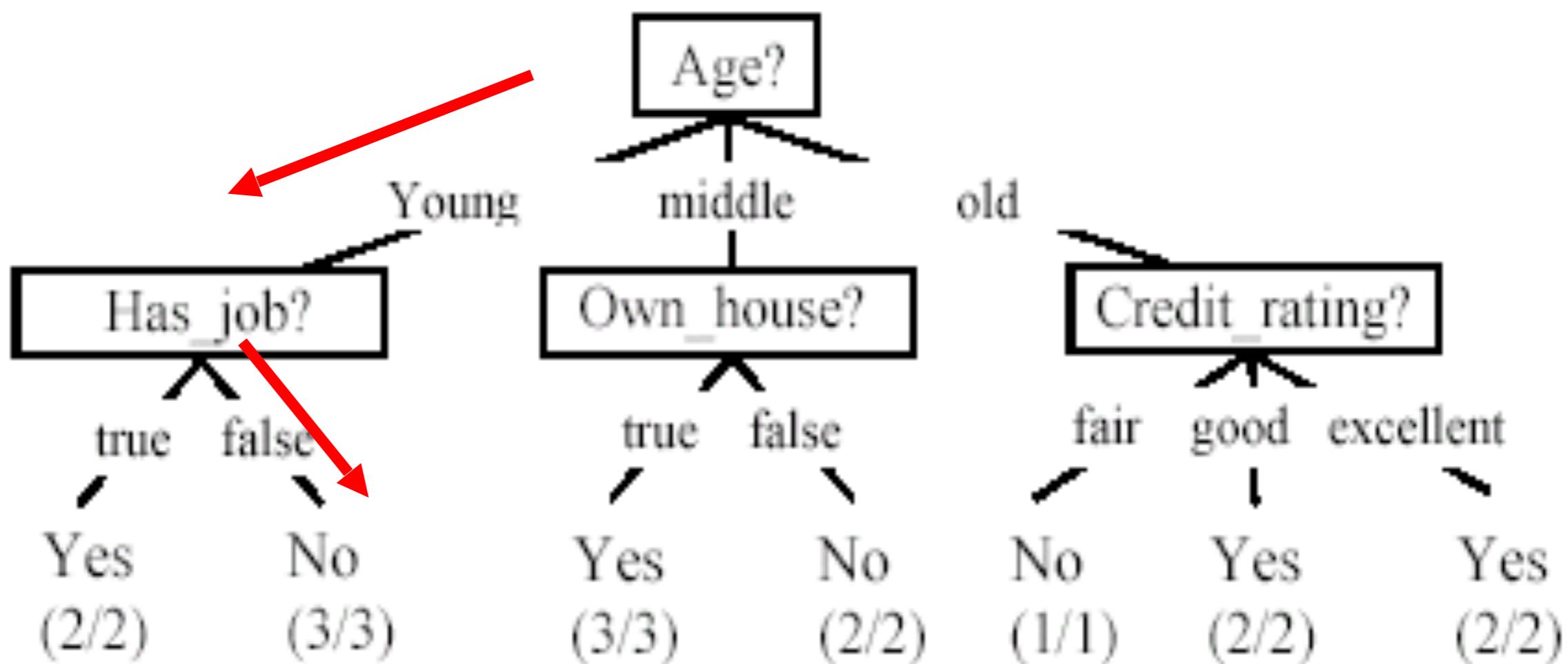
# A decision tree from the loan data

- Decision nodes and leaf nodes (classes)



# Use the decision tree

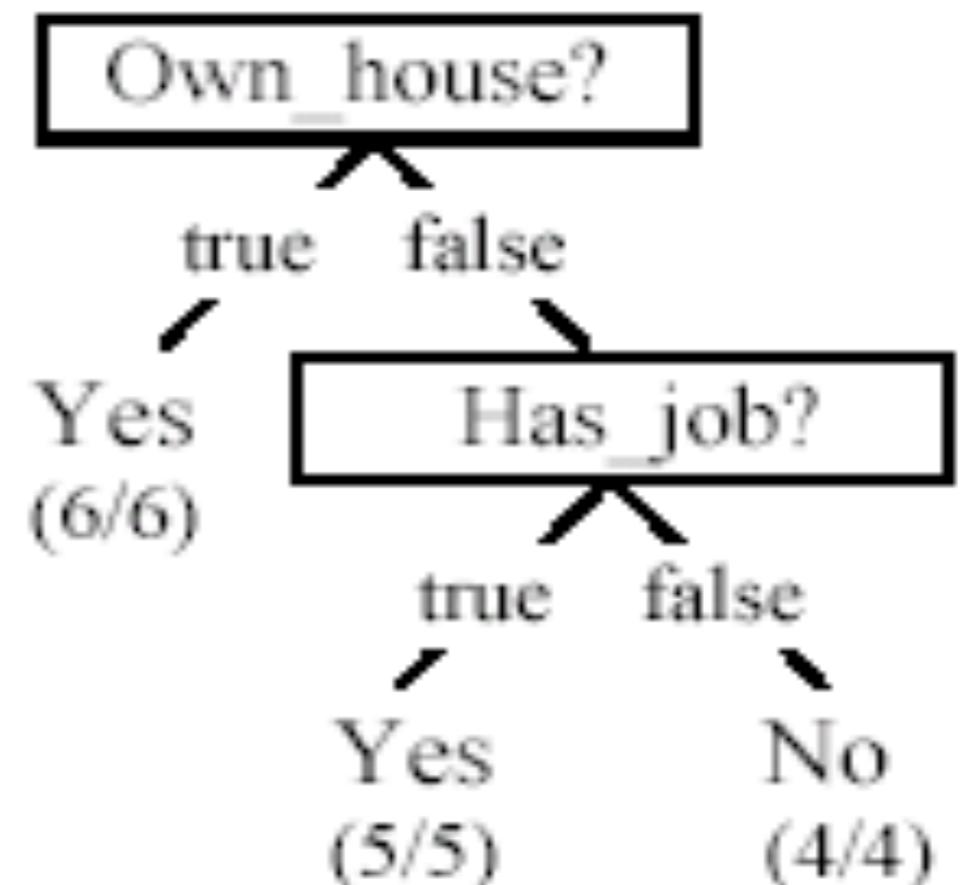
| Age   | Has_Job | Own_house | Credit-Rating | Class |
|-------|---------|-----------|---------------|-------|
| young | false   | false     | good          | No    |



# Is the decision tree unique?

- No. Here is a simpler tree.
- We want **smaller tree** and **accurate tree**.
  - Easy to understand and perform better.

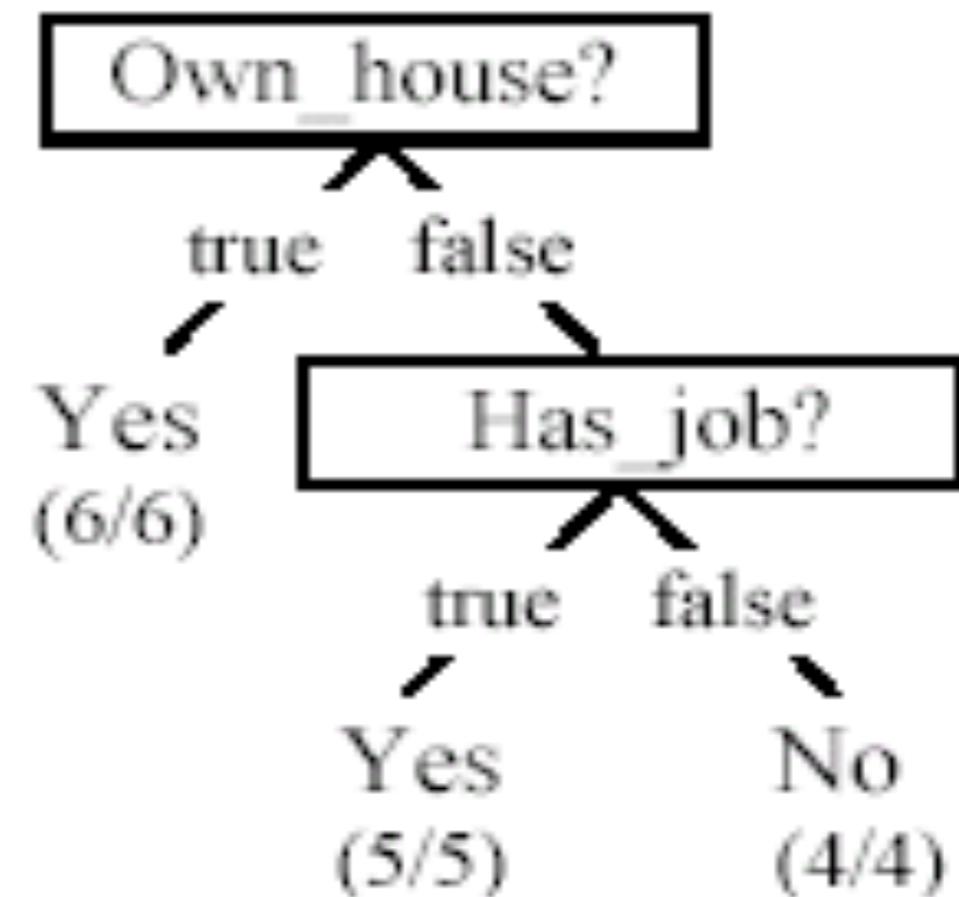
- Finding the best tree is NP-hard.
- All current tree building algorithms are heuristic algorithms



NP-hard (Non-deterministic Polynomial-time hard), in computational complexity theory, is a class of problems that are, informally, "at least as hard as the hardest problems in NP"

# From a decision tree to a set of rules

- A decision tree can be converted to a set of rules
- Each path from the root to a leaf is a rule.



Own\_house = true → Class = Yes [sup=6/15, conf=6/6]

Own\_house = false, Has\_job = true → Class = Yes [sup=5/15, conf=5/5]

Own\_house = false, Has\_job = false → Class = No [sup=4/15, conf=4/4]

# Algorithm for decision tree learning

Basic algorithm (a greedy **divide-and-conquer** algorithm)

- Assume attributes are categorical now (continuous attributes can be handled too)
- Tree is constructed in a **top-down recursive manner**
- At start, all the training examples are at the root
- Examples are partitioned recursively based on selected attributes
- Attributes are selected on the basis of an impurity function (e.g., **information gain**)

Conditions for stopping partitioning

- All examples for a given node belong to the same class
- There are no remaining attributes for further partitioning – majority class is the leaf
- There are no examples left

# Decision tree learning algorithm

```
. Algorithm decisionTree(D, A, T)
1 if D contains only training examples of the same class $c_j \in C$ then
2 make T a leaf node labeled with class c_j ;
3 elseif $A = \emptyset$ then
4 make T a leaf node labeled with c_j , which is the most frequent class in D
5 else // D contains examples belonging to a mixture of classes. We select a single
6 // attribute to partition D into subsets so that each subset is purer
7 $p_0 = \text{impurityEval-1}(D)$;
8 for each attribute $A_i \in \{A_1, A_2, \dots, A_k\}$ do
9 $p_i = \text{impurityEval-2}(A_i, D)$
10 end
11 Select $A_g \in \{A_1, A_2, \dots, A_k\}$ that gives the biggest impurity reduction,
12 computed using $p_0 - p_g$
13 if $p_0 - p_g < \text{threshold}$ then // A_g does not significantly reduce impurity p_0
14 make T a leaf node labeled with c_j , the most frequent class in D .
15 else // A_g is able to reduce impurity p_0
16 Make T a decision node on A_g ;
17 Let the possible values of A_g be v_1, v_2, \dots, v_m . Partition D into m
18 disjoint subsets D_1, D_2, \dots, D_m based on the m values of A_g .
19 for each D_j in $\{D_1, D_2, \dots, D_m\}$ do
20 if $D_j \neq \emptyset$ then
21 create a branch (edge) node T_j for v_j as a child node of T ;
22 decisionTree($D_j, A - \{A_g\}, T_j$) // A_g is removed
23 end
24 end
25 end
26 end
```

# Choose an attribute to partition data

- The *key* to building a decision tree - which attribute to choose in order to branch.
- The objective is to reduce impurity or uncertainty in data as much as possible.

A subset of data is **pure** if all instances belong to the same class.

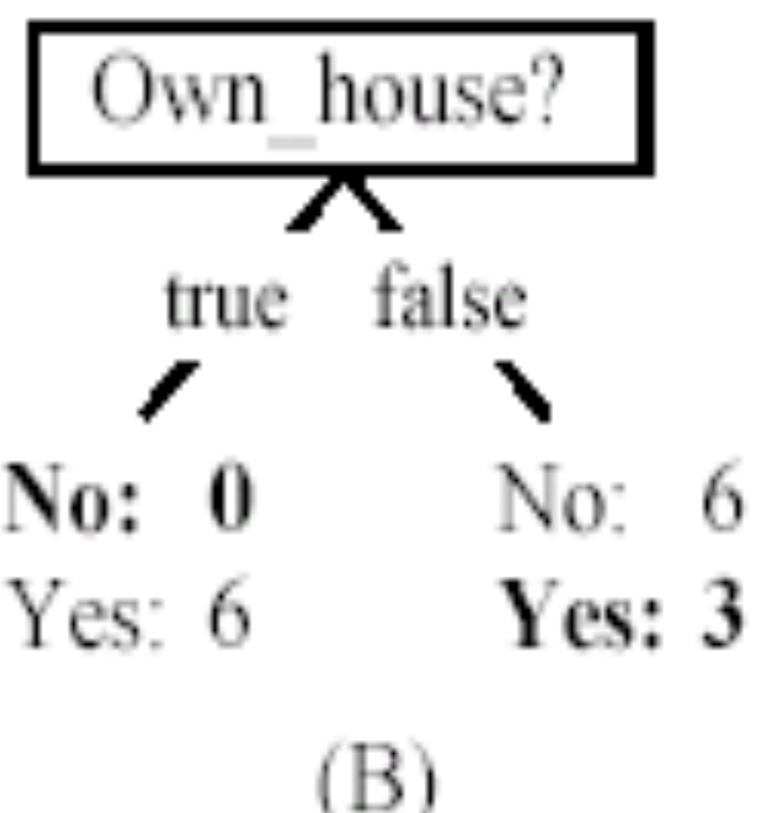
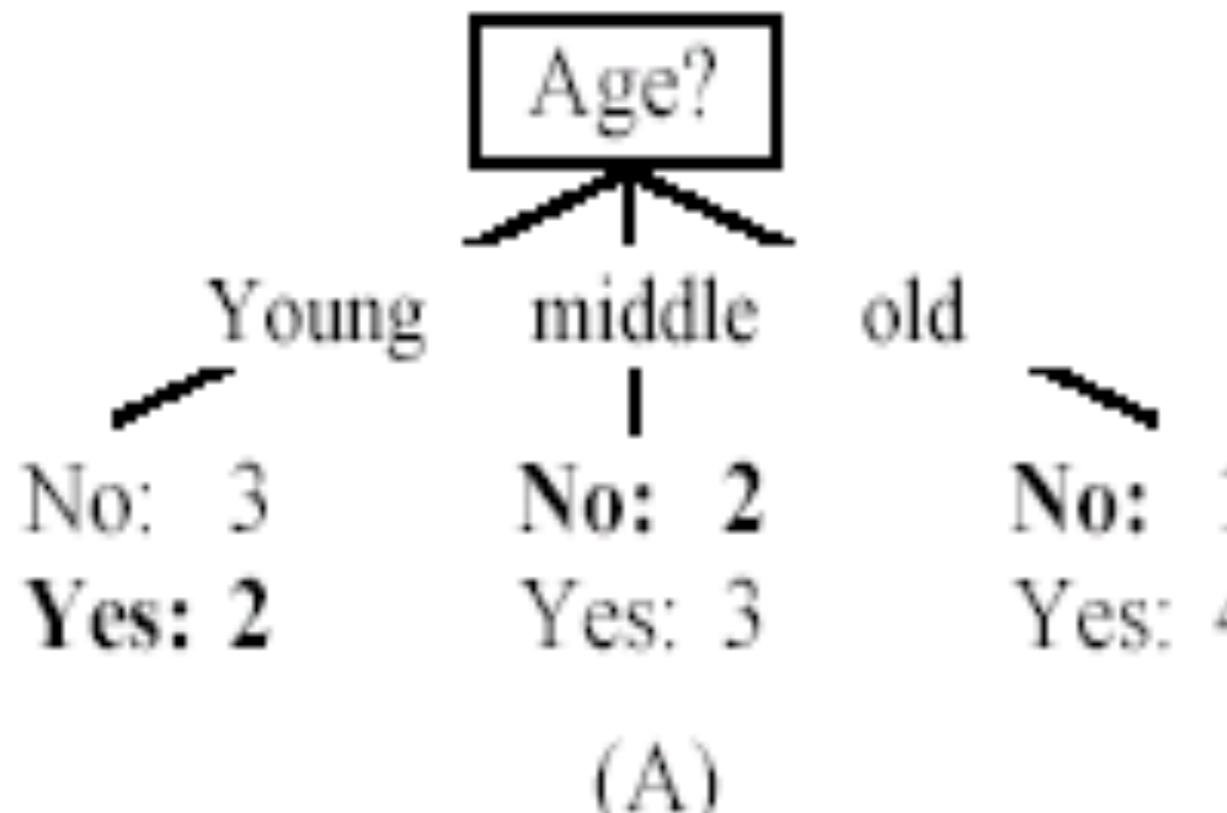
- The *heuristic* in C4.5 is to choose the attribute with the maximum **Information Gain** or **Gain Ratio** based on information theory.

# The loan data (reproduced)

Approved or not

| ID | Age    | Has_Job | Own_House | Credit_Rating | Class |
|----|--------|---------|-----------|---------------|-------|
| 1  | young  | false   | false     | fair          | No    |
| 2  | young  | false   | false     | good          | No    |
| 3  | young  | true    | false     | good          | Yes   |
| 4  | young  | true    | true      | fair          | Yes   |
| 5  | young  | false   | false     | fair          | No    |
| 6  | middle | false   | false     | fair          | No    |
| 7  | middle | false   | false     | good          | No    |
| 8  | middle | true    | true      | good          | Yes   |
| 9  | middle | false   | true      | excellent     | Yes   |
| 10 | middle | false   | true      | excellent     | Yes   |
| 11 | old    | false   | true      | excellent     | Yes   |
| 12 | old    | false   | true      | good          | Yes   |
| 13 | old    | true    | false     | good          | Yes   |
| 14 | old    | true    | false     | excellent     | Yes   |
| 15 | old    | false   | false     | fair          | No    |

# Two possible roots, which is better?



- Fig. (B) seems to be better.

# Information theory

Information theory provides a mathematical basis for measuring the information content.

To understand the notion of information, think about it as providing the answer to a question, for example, whether a coin will come up heads.

- If one already has a good guess about the answer, then the actual answer is less informative.
- If one already knows that the coin is rigged so that it will come with heads with probability 0.99, then a message (advanced information) about the actual outcome of a flip is worth less than it would be for a honest coin (50-50).

# Information theory (cont ...)

- For a fair (honest) coin, you have no information, and you are willing to pay more (say in terms of \$) for advanced information - less you know, the more valuable the information.
- **Information theory** uses this same intuition, but instead of measuring the value for information in dollars, it measures information contents in **bits**.
- One bit of information is enough to answer a yes/no question about which one has no idea, such as the flip of a fair coin

# Information theory: Entropy measure

The entropy formula,

$$\text{entropy}(D) = - \sum_{j=1}^{|C|} \Pr(c_j) \log_2 \Pr(c_j)$$

$$\sum_{j=1}^{|C|} \Pr(c_j) = 1,$$

$\Pr(c_j)$  is the probability of class  $c_j$  in data set  $D$

We use entropy as a **measure of impurity or disorder** of data set  $D$ . (Or, a measure of information in a tree)

# Entropy measure: let us get a feeling

1. The data set  $D$  has 50% positive examples ( $\Pr(\text{positive}) = 0.5$ ) and 50% negative examples ( $\Pr(\text{negative}) = 0.5$ ).

$$\text{entropy}(D) = -0.5 \times \log_2 0.5 - 0.5 \times \log_2 0.5 = 1$$

2. The data set  $D$  has 20% positive examples ( $\Pr(\text{positive}) = 0.2$ ) and 80% negative examples ( $\Pr(\text{negative}) = 0.8$ ).

$$\text{entropy}(D) = -0.2 \times \log_2 0.2 - 0.8 \times \log_2 0.8 = 0.722$$

3. The data set  $D$  has 100% positive examples ( $\Pr(\text{positive}) = 1$ ) and no negative examples, ( $\Pr(\text{negative}) = 0$ ).

$$\text{entropy}(D) = -1 \times \log_2 1 - 0 \times \log_2 0 = 0$$

- As the data become purer and purer, the entropy value becomes smaller and smaller. This is useful to us!

# Information gain

Given a set of examples  $D$ , we first compute its entropy:

$$\text{entropy}(D) = - \sum_{j=1}^{|C|} \Pr(c_j) \log_2 \Pr(c_j)$$

If we make attribute  $A_i$ , with  $v$  values, the root of the current tree, this will partition  $D$  into  $v$  subsets  $D_1, D_2, \dots, D_v$ . The expected entropy if  $A_i$  is used as the current root:

$$\text{entropy}_{A_i}(D) = \sum_{j=1}^v \frac{|D_j|}{|D|} \times \text{entropy}(D_j)$$

# Information gain (cont...)

Information gained by selecting attribute  $A_i$  to branch or to partition the data is

$$gain(D, A_i) = entropy(D) - entropy_{A_i}(D)$$

We choose the attribute with the highest gain to branch/split the current tree.

# An example

$$\text{entropy}(D) = \frac{6}{15} \times \log_2 \frac{6}{15} + \frac{9}{15} \times \log_2 \frac{9}{15} = 0.971$$

$$\begin{aligned}\text{entropy}_{\text{Own\_house}}(D) &= \frac{6}{15} \times \text{entropy}(D_1) + \frac{9}{15} \times \text{entropy}(D_2) \\ &= \frac{6}{15} \times 0 + \frac{9}{15} \times 0.918 \\ &= 0.551\end{aligned}$$

$$\begin{aligned}\text{entropy}_{\text{Age}}(D) &= \frac{5}{15} \times \text{entropy}(D_1) + \frac{5}{15} \times \text{entropy}(D_2) + \frac{5}{15} \times \text{entropy}(D_3) \\ &= \frac{5}{15} \times 0.971 + \frac{5}{15} \times 0.971 + \frac{5}{15} \times 0.722 \\ &= 0.888\end{aligned}$$

| ID | Age    | Has_Job | Own_House | Credit_Rating | Class |
|----|--------|---------|-----------|---------------|-------|
| 1  | young  | false   | false     | fair          | No    |
| 2  | young  | false   | false     | excellent     | No    |
| 3  | young  | true    | false     | good          | Yes   |
| 4  | young  | true    | true      | good          | Yes   |
| 5  | young  | false   | false     | fair          | No    |
| 6  | middle | false   | false     | fair          | No    |
| 7  | middle | false   | false     | good          | No    |
| 8  | middle | true    | true      | good          | Yes   |
| 9  | middle | false   | true      | excellent     | Yes   |
| 10 | middle | false   | true      | excellent     | Yes   |
| 11 | old    | false   | true      | excellent     | Yes   |
| 12 | old    | false   | true      | good          | Yes   |
| 13 | old    | true    | false     | good          | Yes   |
| 14 | old    | true    | false     | excellent     | Yes   |
| 15 | old    | false   | false     | fair          | No    |

| Age    | Yes | No | entropy(Di) |
|--------|-----|----|-------------|
| young  | 2   | 3  | 0.971       |
| middle | 3   | 2  | 0.971       |
| old    | 4   | 1  | 0.722       |

$$gain(D, \text{Age}) = 0.971 - 0.888 = 0.083$$

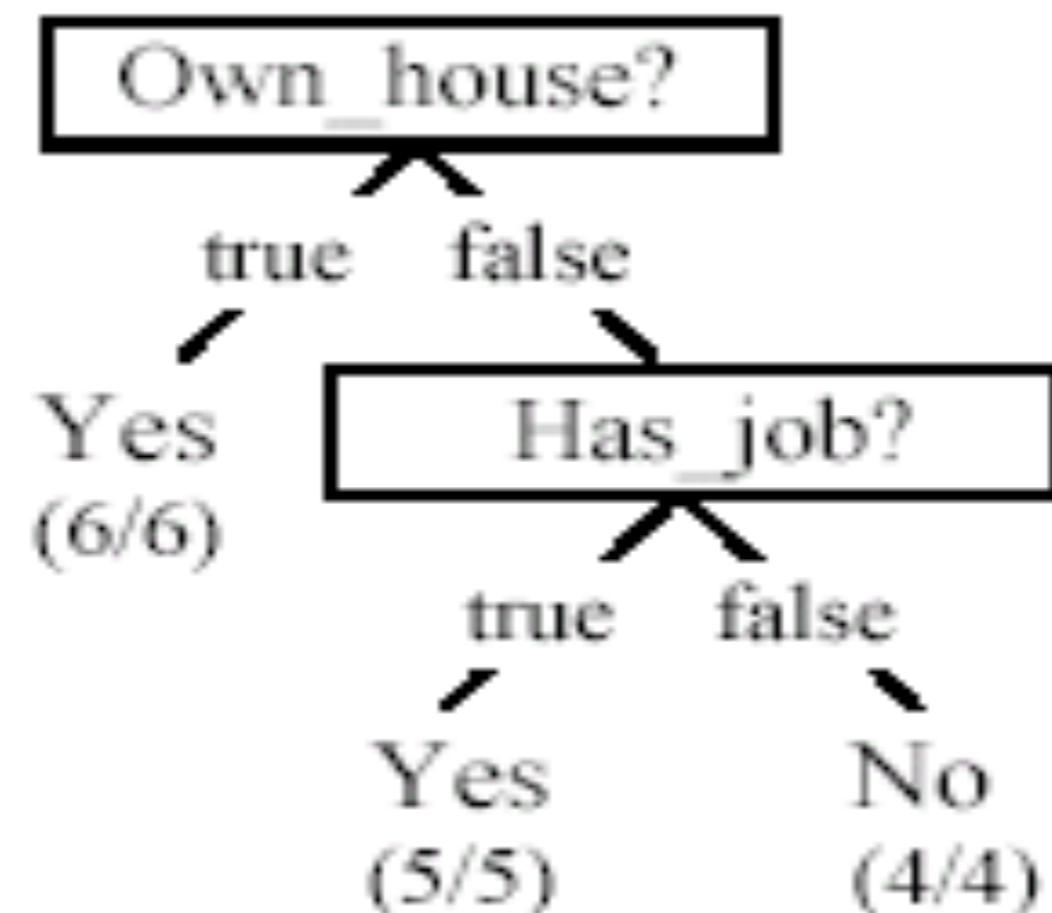
$$gain(D, \text{Own\_house}) = 0.971 - 0.551 = 0.420$$

$$gain(D, \text{Has_Job}) = 0.971 - 0.647 = 0.324$$

$$gain(D, \text{Credit_Rating}) = 0.971 - 0.608 = 0.363$$

- Own\_house is the best choice for the root.

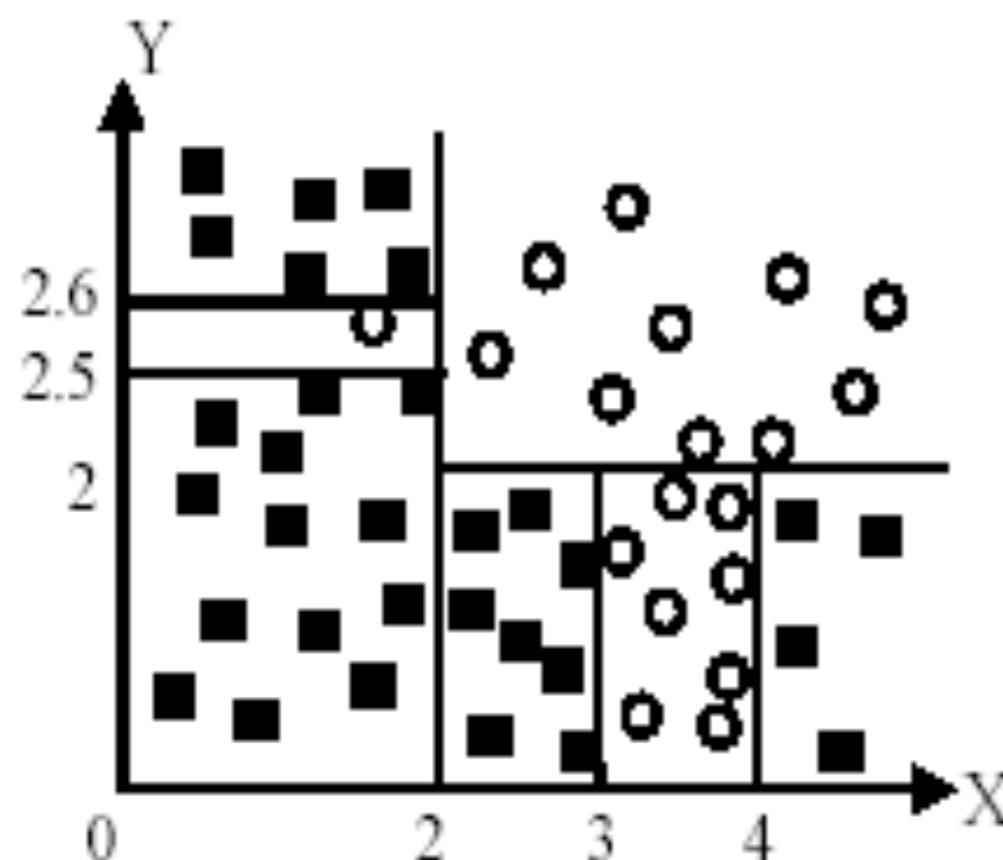
# We build the final tree



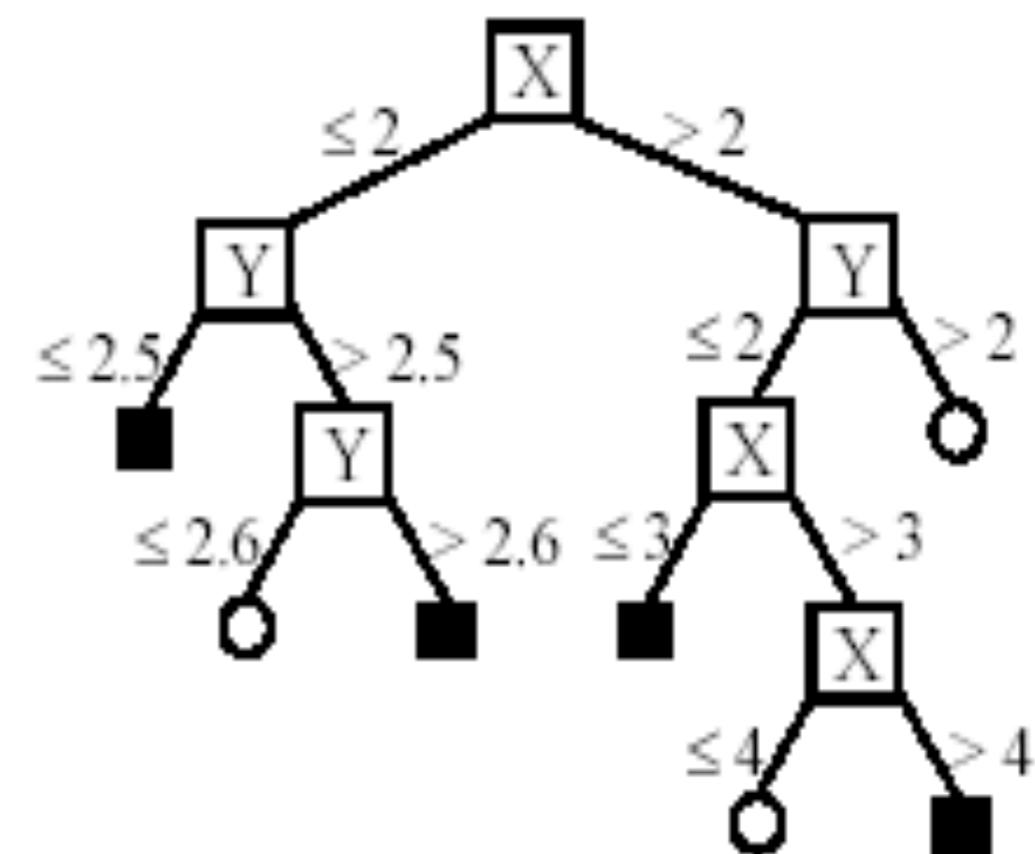
# Handling continuous attributes

- Handle continuous attribute by splitting into two intervals (can be more) at each node.
- How to find the best threshold to divide?
  - Use information gain or gain ratio again
  - Sort all the values of an continuous attribute in increasing order  $\{v_1, v_2, \dots, v_r\}$ ,
  - One possible threshold between two adjacent values  $v_i$  and  $v_{i+1}$ . Try all possible thresholds and find the one that maximizes the gain (or gain ratio).

# An example in a continuous space



(A) A partition of the data space



(B). The decision tree

# Avoid overfitting in classification

**Overfitting:** A tree may overfit the training data

Good accuracy on training data but poor on test data

Symptoms: tree too deep and too many branches, some may reflect anomalies due to noise or outliers

Two approaches to avoid overfitting

**Pre-pruning:** Halt tree construction early

Difficult to decide because we do not know what may happen subsequently if we keep growing the tree.

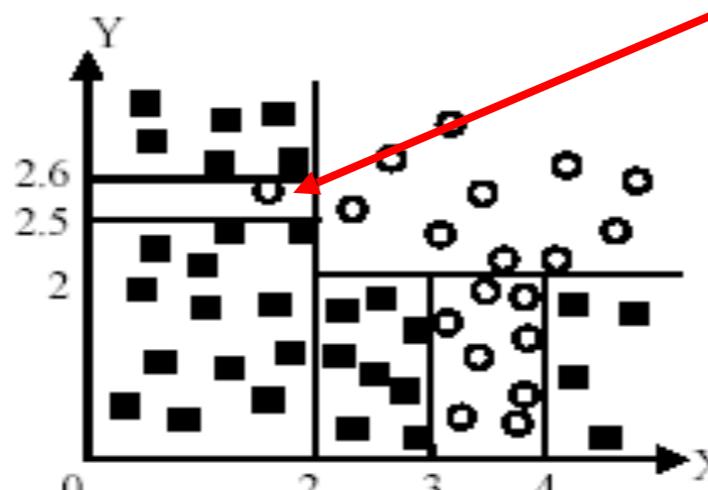
**Post-pruning:** Remove branches or sub-trees from a “fully grown” tree.

This method is commonly used. C4.5 uses a statistical method to estimates the errors at each node for pruning.

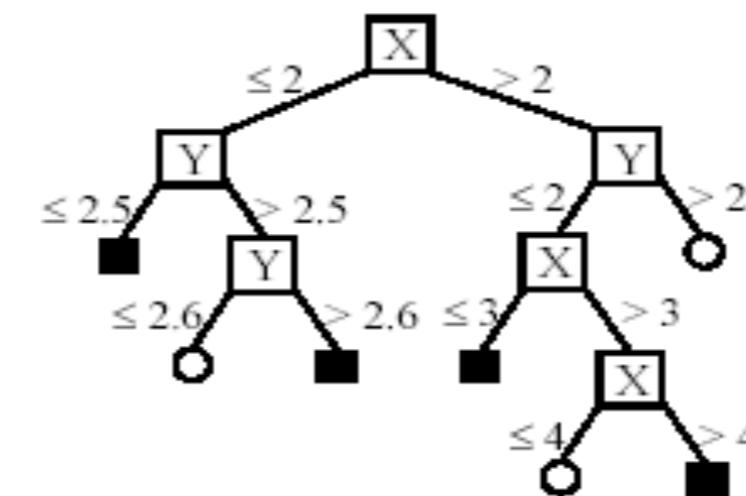
A validation set may be used for pruning as well.

# An example

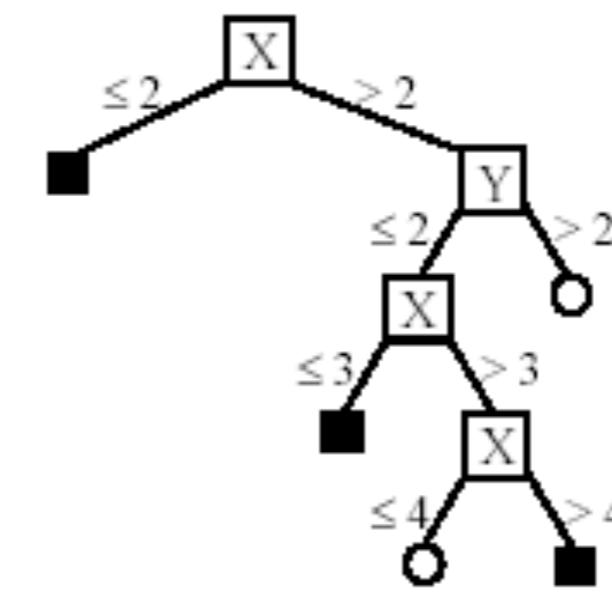
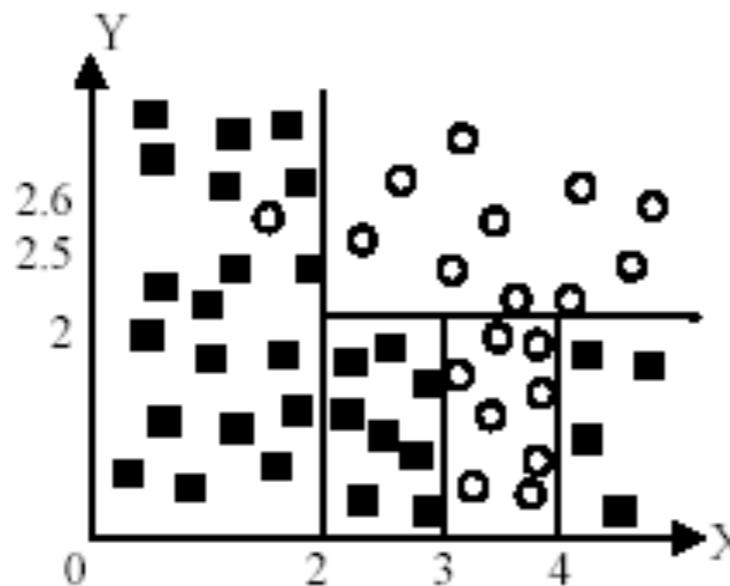
## Likely to overfit the data



(A) A partition of the data space



#### (B). The decision tree



# Evaluating classification methods

## Predictive accuracy

$$Accuracy = \frac{\text{Number of correct classifications}}{\text{Total number of test cases}}$$

## Efficiency

time to construct the model

time to use the model

**Robustness:** handling noise and missing values

**Scalability:** efficiency in disk-resident databases

**Interpretability:**

understandable and insight provided by the model

**Compactness of the model:** size of the tree, or the number of rules.

# Evaluation methods

- **Holdout set:** The available data set  $D$  is divided into two disjoint subsets,
  - the *training set*  $D_{train}$  (for learning a model)
  - the *test set*  $D_{test}$  (for testing the model)
- **Important:** training set should not be used in testing and the test set should not be used in learning.
  - Unseen test set provides a unbiased estimate of accuracy.
- The test set is also called the **holdout set**. (the examples in the original data set  $D$  are all labeled with classes.)
- This method is mainly used when the data set  $D$  is large.

# Evaluation methods (cont...)

- **n-fold cross-validation:** The available data is partitioned into  $n$  equal-size disjoint subsets.
- Use each subset as the test set and combine the rest  $n-1$  subsets as the training set to learn a classifier.
- The procedure is run  $n$  times, which give  $n$  accuracies.
- The final estimated accuracy of learning is the average of the  $n$  accuracies.
- 10-fold and 5-fold cross-validations are commonly used.
- This method is used when the available data is not large.

# Evaluation methods (cont...)

- **Leave-one-out cross-validation**: This method is used when the data set is very small.
- It is a special case of cross-validation
- Each fold of the cross validation has only **a single test example** and all the rest of the data is used in training.
- If the original data has  $m$  examples, this is  **$m$ -fold cross-validation**

# Evaluation methods (cont...)

- **Validation set:** the available data is divided into three subsets,
  - a training set,
  - a validation set and
  - a test set.
- A validation set is used frequently for estimating parameters in learning algorithms.
- In such cases, the values that give the best accuracy on the validation set are used as the final parameter values.
- Cross-validation can be used for parameter estimating as well.

# Classification measures

- Accuracy is only one measure (error = 1-accuracy).
- **Accuracy is not suitable in some applications.**
- In text mining, we may only be interested in the documents of a particular topic, which are only a small portion of a big document collection.
- In classification involving skewed or highly imbalanced data, e.g., network intrusion and financial fraud detections, **we are interested only in the minority class.**
  - High accuracy does not mean any intrusion is detected.
  - E.g., 1% intrusion. Achieve 99% accuracy by doing nothing.
- The class of interest is commonly called the **positive class**, and the rest **negative classes**.

# Precision and recall measures

- Used in information retrieval and text classification.
- We use a confusion matrix to introduce them.

|                 | Classified Positive | Classified Negative |
|-----------------|---------------------|---------------------|
| Actual Positive | TP                  | FN                  |
| Actual Negative | FP                  | TN                  |

where

*TP*: the number of correct classifications of the positive examples (**true positive**),

*FN*: the number of incorrect classifications of positive examples (**false negative**),

*FP*: the number of incorrect classifications of negative examples (**false positive**), and

*TN*: the number of correct classifications of negative examples (**true negative**).

# Precision and recall measures (cont...)

|                 | Classified Positive | Classified Negative |
|-----------------|---------------------|---------------------|
| Actual Positive | TP                  | FN                  |
| Actual Negative | FP                  | TN                  |

$$p = \frac{TP}{TP + FP}.$$
      
$$r = \frac{TP}{TP + FN}.$$

- **Precision  $p$**  is the number of correctly classified positive examples divided by the total number of examples that are classified as positive.
- **Recall  $r$**  is the number of correctly classified positive examples divided by the total number of actual positive examples in the test set.

# An example

|                 | Classified Positive | Classified Negative |
|-----------------|---------------------|---------------------|
| Actual Positive | 1                   | 99                  |
| Actual Negative | 0                   | 1000                |

This confusion matrix gives

precision  $p = 100\%$  and

recall  $r = 1\%$

because we only classified one positive example correctly and no negative examples wrongly.

Note: precision and recall only measure classification on the positive class.

# $F_1$ -value (also called $F_1$ -score)

It is hard to compare two classifiers using two measures.  $F_1$  score combines precision and recall into one measure

$$F_1 = \frac{2pr}{p+r}$$

$F_1$ -score is the harmonic mean of precision and recall.

$$F_1 = \frac{2}{\frac{1}{p} + \frac{1}{r}}$$

The harmonic mean of two numbers tends to be closer to the smaller of the two.

For  $F_1$ -value to be large, both  $p$  and  $r$  must be large.

# Receive operating characteristics curve

It is commonly called the **ROC curve**.

It is a plot of the **true positive rate (TPR)** against the **false positive rate (FPR)**.

**True positive rate:**

$$TPR = \frac{TP}{TP + FN}$$

**False positive rate:**

$$FPR = \frac{FP}{TN + FP}$$

# Sensitivity and Specificity

In statistics, there are two other evaluation measures:

**Sensitivity**: Same as TPR

**Specificity**: Also called **True Negative Rate (TNR)**

Then we have

$$TNR = \frac{TN}{TN + FP}$$

$$FPR = 1 - specificity$$

# Example ROC curves

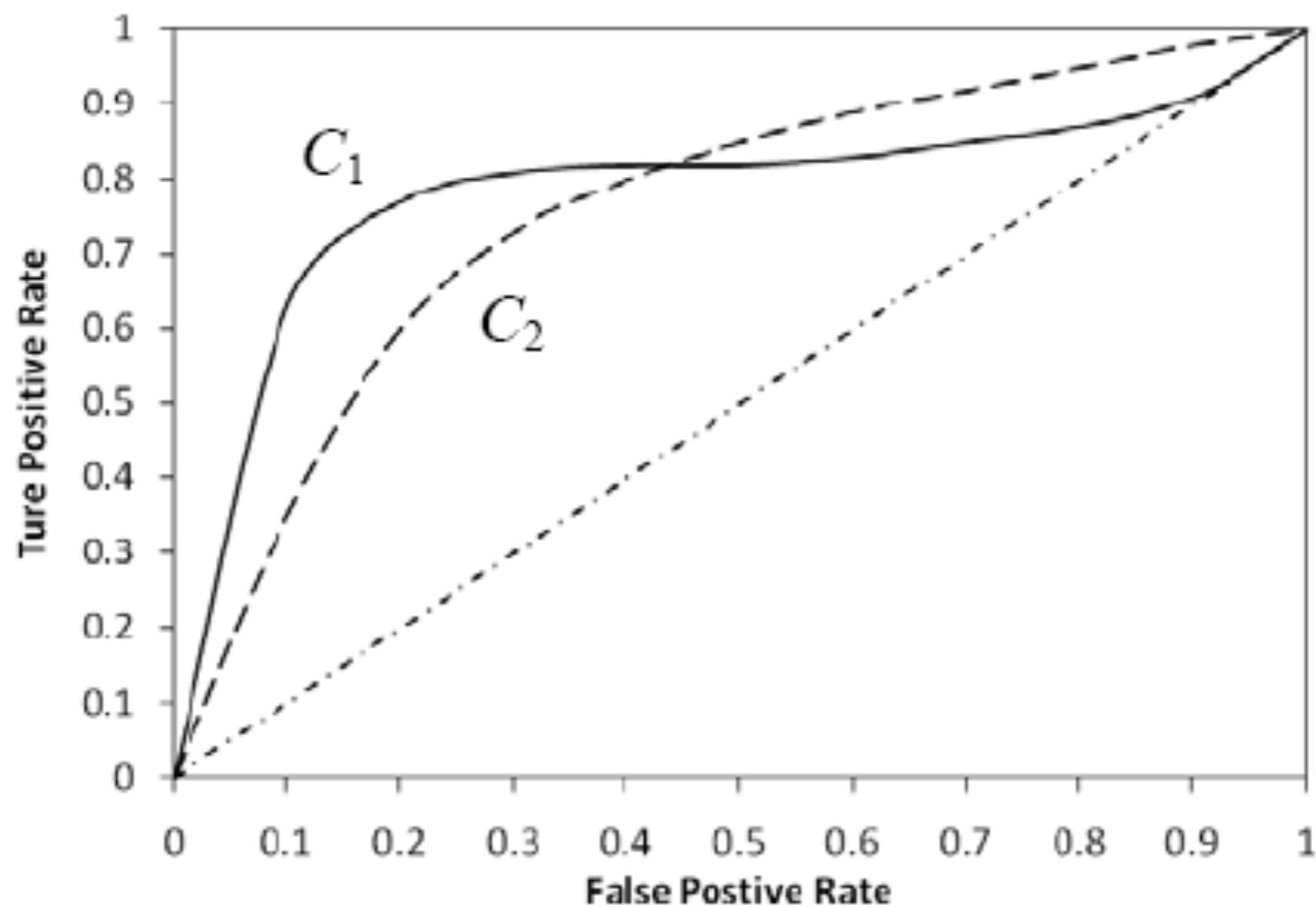


Fig. 3.8. ROC curves for two classifiers ( $C_1$  and  $C_2$ ) on the same data

# Area under the curve (AUC)

Which classifier is better,  $C_1$  or  $C_2$ ?

It depends on which region you talk about.

Can we have one measure?

Yes, we compute the area under the curve (AUC)

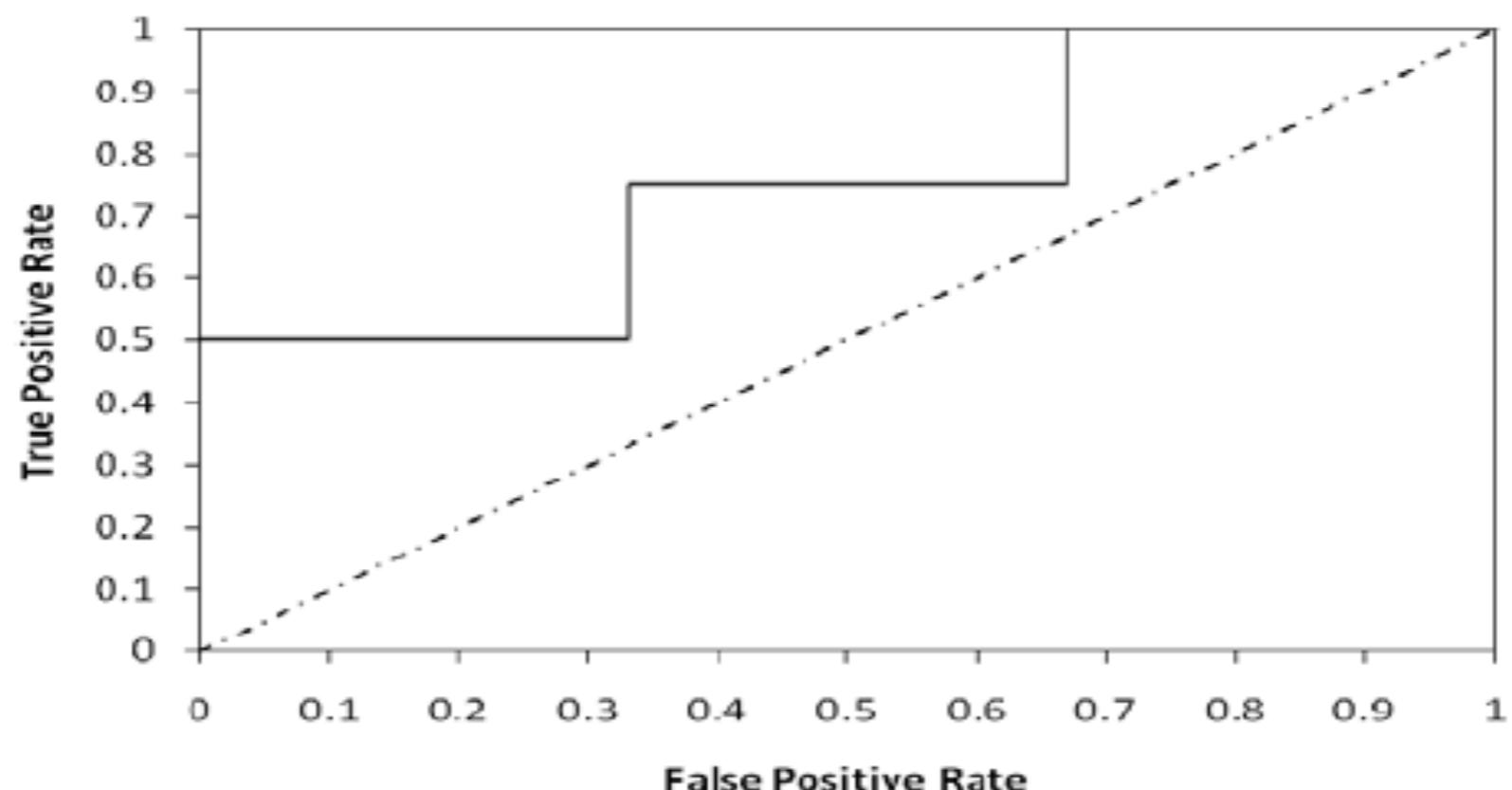
If AUC for  $C_i$  is greater than that of  $C_j$ , it is said that  $C_i$  is better than  $C_j$ .

If a classifier is perfect, its AUC value is 1

If a classifier makes all random guesses, its AUC value is 0.5.

# Drawing an ROC curve

| Rank         | 1    | 2   | 3   | 4    | 5    | 6    | 7    | 8    | 9    | 10   |
|--------------|------|-----|-----|------|------|------|------|------|------|------|
| Actual class | +    | +   | -   | -    | +    | -    | -    | +    | -    | -    |
| TP           | 1    | 2   | 2   | 2    | 3    | 3    | 3    | 4    | 4    | 4    |
| FP           | 0    | 0   | 0   | 1    | 2    | 2    | 3    | 4    | 4    | 5    |
| TN           | 6    | 6   | 6   | 5    | 4    | 4    | 3    | 2    | 1    | 0    |
| FN           | 4    | 3   | 2   | 2    | 2    | 1    | 1    | 0    | 0    | 0    |
| TPR          | 0.25 | 0.5 | 0.5 | 0.5  | 0.75 | 0.75 | 0.75 | 1    | 1    | 1    |
| FPR          | 0    | 0   | 0   | 0.17 | 0.33 | 0.33 | 0.50 | 0.67 | 0.67 | 0.83 |



# Another evaluation method: Scoring and ranking

- **Scoring** is related to classification.
- We are interested in a single class (**positive class**), e.g., buyers class in a marketing database.
- Instead of assigning each test instance a definite class, scoring assigns a probability estimate (PE) to indicate the likelihood that the example belongs to the positive class.

# Ranking and lift analysis

- After each example is given a PE score, we can rank all examples according to their PEs.
- We then divide the data into  $n$  (say 10) bins. A lift curve can be drawn according how many positive examples are in each bin. This is called **lift analysis**.
- Classification systems can be used for scoring. Need to produce a probability estimate.
  - E.g., in decision trees, we can use the confidence value at each leaf node as the score.

# An example

- We want to send promotion materials to potential customers to sell a watch.
- Each package cost \$0.50 to send (material and postage).
- If a watch is sold, we make \$5 profit.
- Suppose we have a large amount of past data for building a predictive/classification model. We also have a large list of potential customers.
- How many packages should we send and who should we send to?

# An example

Assume that the test set has 10000 instances. Out of this, 500 are positive cases.

After the classifier is built, we score each test instance. We then rank the test set, and divide the ranked test set into 10 bins.

**Each bin has 1000 test instances.**

Bin 1 has 210 actual positive instances

Bin 2 has 120 actual positive instances

Bin 3 has 60 actual positive instances

...

Bin 10 has 5 actual positive instances

# Lift curve

Bin 1

2

3

4

5

6

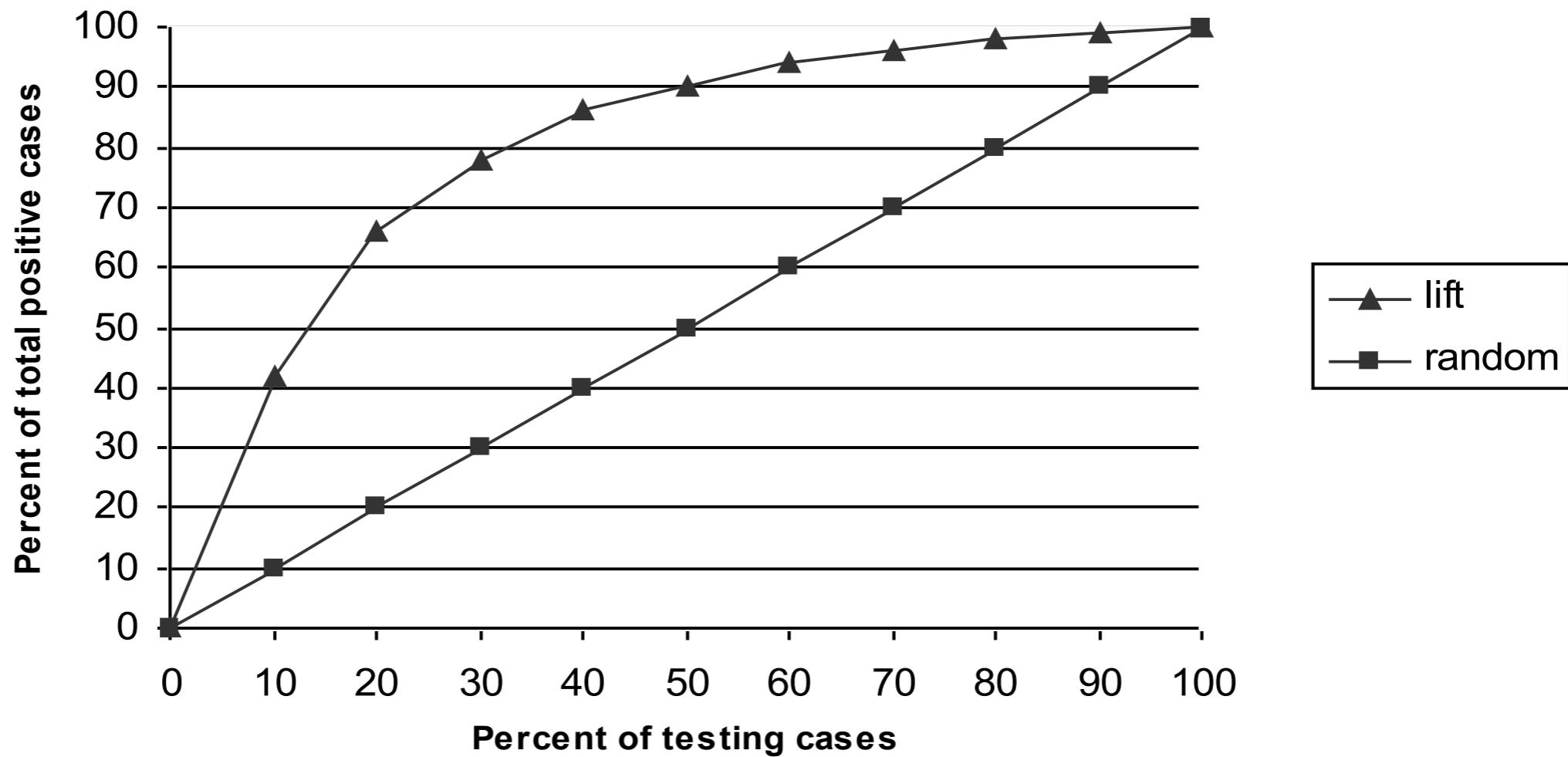
7

8

9

10

|     |     |     |     |        |       |        |        |       |      |
|-----|-----|-----|-----|--------|-------|--------|--------|-------|------|
| 210 | 120 | 60  | 40  | 22     | 18    | 12     | 7      | 6     | 5    |
| 42% | 24% | 12% | 8%  | 4.40%  | 3.60% | 2.40%  | 1.40%  | 1.20% | 1%   |
| 42% | 66% | 78% | 86% | 90.40% | 94%   | 96.40% | 97.80% | 99%   | 100% |



# Topic

Basic concepts

Decision tree induction

Evaluation of classifiers

## **Classification using association rules**

Naïve Bayesian classification

Naïve Bayes for text classification

K-nearest neighbor

Ensemble methods: Bagging and Boosting

Summary

# Three approaches

- Three main approaches of using association rules for classification.
  - Using class association rules to build classifiers
  - Using class association rules as attributes/features
  - Using normal association rules for classification

# Using Class Association Rules

**Classification:** mine a small set of rules existing in the data to form a classifier or predictor.

It has a target attribute: **Class attribute**

**Association rules:** have no fixed target, but we can fix a target.

**Class association rules (CAR):** has a target class attribute. E.g.,

Own\_house = true  $\rightarrow$  Class =Yes [sup=6/15, conf=6/6]

CARs can obviously be used for classification.

# Decision tree vs. CARs

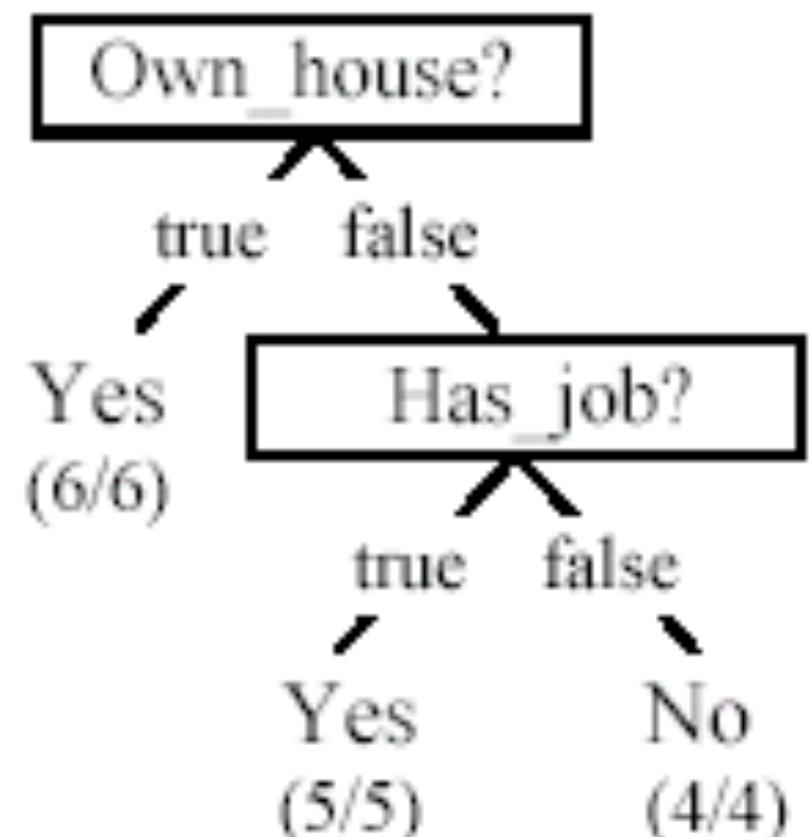
The decision tree below generates the following 3 rules.

Own\_house = true → Class = Yes [sup=6/15, conf=6/6]

Own\_house = false, Has\_job = true → Class=Yes [sup=5/15, conf=5/5]

Own\_house = false, Has\_job = false → Class=No [sup=4/15, conf=4/4]

- But there are many other rules that are not found by the decision tree



# There are many more rules

|                                         |                      |
|-----------------------------------------|----------------------|
| Age = young, Has_job = true → Class=Yes | [sup=2/15, conf=2/2] |
| Age = young, Has_job = false → Class=No | [sup=3/15, conf=3/3] |
| Credit_Rating = fair → Class=No         | [sup=4/15, conf=4/4] |
| Credit_Rating = good → Class=Yes        | [sup=5/15, conf=5/6] |

and many more, if we use minsup = 2/15 = 13.3% and minconf = 80%. ■

CAR mining finds all of them.

In many cases, rules not in the decision tree (or a rule list) may perform classification better. Such rules may also be actionable in practice

| ID | Age    | Has_Job | Own_House | Credit_Rating | Class |
|----|--------|---------|-----------|---------------|-------|
| 1  | young  | false   | false     | fair          | No    |
| 2  | young  | false   | false     | excellent     | No    |
| 3  | young  | true    | false     | good          | Yes   |
| 4  | young  | true    | true      | good          | Yes   |
| 5  | young  | false   | false     | fair          | No    |
| 6  | middle | false   | false     | fair          | No    |
| 7  | middle | false   | false     | good          | No    |
| 8  | middle | true    | true      | good          | Yes   |
| 9  | middle | false   | true      | excellent     | Yes   |
| 10 | middle | false   | true      | excellent     | Yes   |
| 11 | old    | false   | true      | excellent     | Yes   |
| 12 | old    | false   | true      | good          | Yes   |
| 13 | old    | true    | false     | good          | Yes   |
| 14 | old    | true    | false     | excellent     | Yes   |
| 15 | old    | false   | false     | fair          | No    |

# Decision tree vs. CARs (cont ...)

Association mining require discrete attributes. Decision tree learning uses both discrete and continuous attributes.

**CAR mining requires continuous attributes discretized. There are several such algorithms.**

Decision tree is not constrained by minsup or minconf, and thus is able to find rules with very low support. Of course, such rules may be pruned due to the possible overfitting.

# Considerations in CAR mining

## Multiple minimum class supports

Deal with imbalanced class distribution, e.g., some class is rare, 98% negative and 2% positive.

We can set the  $\text{minsup}(\text{positive}) = 0.2\%$  and  $\text{minsup}(\text{negative}) = 2\%$ .

If we are not interested in classification of negative class, we may not want to generate rules for negative class. We can set  $\text{minsup}(\text{negative})=100\%$  or more.

**Rule pruning** may be performed.

# Building classifiers

There are many ways to build classifiers using CARs. Several existing systems available.

**Strongest rules:** After CARs are mined, do nothing.

For each test case, we simply choose the most confident rule that covers the test case to classify it. Microsoft SQL Server has a similar method.

Or, using a combination of rules.

**Selecting a subset of Rules**

used in the CBA system.

similar to sequential covering.

# CBA: Rules are sorted first

**Definition:** Given two rules,  $r_i$  and  $r_j$ ,  $r_i \sqsupseteq r_j$  (also called  $r_i$  precedes  $r_j$  or  $r_i$  has a higher precedence than  $r_j$ ) if

- the confidence of  $r_i$  is greater than that of  $r_j$ , or
- their confidences are the same, but the support of  $r_i$  is greater than that of  $r_j$ , or
- both the confidences and supports of  $r_i$  and  $r_j$  are the same, but  $r_i$  is generated earlier than  $r_j$ .

A CBA classifier  $L$  is of the form:

$$L = \langle r_1, r_2, \dots, r_k, \text{default-class} \rangle$$

# Classifier building using CARs

**Algorithm** CBA( $S, D$ )

- 1  $S = \text{sort}(S)$ ; // sorting is done according to the precedence  $\succ$
- 2  $\text{RuleList} = \emptyset$ ; // the rule list classifier
- 3 **for** each rule  $r \in S$  in sequence **do**
- 4     **if**  $D \neq \emptyset$  AND  $r$  classifies at least one example in  $D$  correctly **then**
- 5         delete from  $D$  all training examples covered by  $r$ ;
- 6         add  $r$  at the end of  $\text{RuleList}$
- 7     **end**
- 8 **end**
- 9 add the majority class as the default class at the end of  $\text{RuleList}$

# Using rules as features

Most classification methods do not fully explore multi-attribute correlations, e.g., naïve Bayesian, decision trees, rules induction, etc.

This method creates extra attributes to augment the original data by

- Using the conditional parts of rules

- Each rule forms an new attribute

- If a data record satisfies the condition of a rule, the attribute value is 1, and 0 otherwise

One can also use only rules as attributes

- Throw away the original data

# Using normal association rules for classification

A widely used approach

Main approach: strongest rules

Main application

Recommendation systems in e-commerce Web site (e.g., amazon.com).

Each rule consequent is the recommended item.

Major advantage: any item can be predicted.

Main issue:

Coverage: rare item rules are not found using classic algo.

Multiple min supports and support difference constraint help a great deal.

# Topic

Basic concepts

Decision tree induction

Evaluation of classifiers

Classification using association rules

## **Naïve Bayesian classification**

Naïve Bayes for text classification

K-nearest neighbor

Ensemble methods: Bagging and Boosting

Summary

# Bayesian classification

**Probabilistic view:** Supervised learning can naturally be studied from a probabilistic point of view.

Let  $A_1$  through  $A_k$  be attributes with discrete values. The class is  $C$ .

Given a test example  $d$  with observed attribute values  $a_1$  through  $a_k$ .

Classification is basically to compute the following posteriori probability.

The prediction is the class  $c_j$  such that

$$\Pr(C = c_j \mid A_1 = a_1, \dots, A_{|A|} = a_{|A|})$$

is maximal

# Apply Bayes' Rule

$$\Pr(C = c_j \mid A_1 = a_1, \dots, A_{|A|} = a_{|A|})$$
$$= \frac{\Pr(A_1 = a_1, \dots, A_{|A|} = a_{|A|} \mid C = c_j) \Pr(C = c_j)}{\Pr(A_1 = a_1, \dots, A_{|A|} = a_{|A|})}$$
$$= \frac{\Pr(A_1 = a_1, \dots, A_{|A|} = a_{|A|} \mid C = c_j) \Pr(C = c_j)}{\sum_{r=1}^{|C|} \Pr(A_1 = a_1, \dots, A_{|A|} = a_{|A|} \mid C = c_r) \Pr(C = c_r)}$$

- $\Pr(C=c_j)$  is the class *prior* probability: easy to estimate from the training data.

# Computing probabilities

The denominator  $P(A_1=a_1, \dots, A_k=a_k)$  is irrelevant for decision making since it is the same for every class.

We only need  $P(A_1=a_1, \dots, A_k=a_k | C=c_i)$ , which can be written as

$$\Pr(A_1=a_1 | A_2=a_2, \dots, A_k=a_k, C=c_j) * \Pr(A_2=a_2, \dots, A_k=a_k | C=c_j)$$

Recursively, the second factor above can be written in the same way, and so on.

Now an assumption is needed.

# Conditional independence

All attributes are conditionally independent given the class  $C = c_j$ .

Formally, we assume,

$$\Pr(A_1=a_1 \mid A_2=a_2, \dots, A_{|A|}=a_{|A|}, C=c_j) = \Pr(A_1=a_1 \mid C=c_j)$$

and so on for  $A_2$  through  $A_{|A|}$ . I.e.,

$$\Pr(A_1 = a_1, \dots, A_{|A|} = a_{|A|} \mid C = c_i) = \prod_{i=1}^{|A|} \Pr(A_i = a_i \mid C = c_j)$$

# Final naïve Bayesian classifier

$$\Pr(C = c_j \mid A_1 = a_1, \dots, A_{|A|} = a_{|A|}) \\ = \frac{\Pr(C = c_j) \prod_{i=1}^{|A|} \Pr(A_i = a_i \mid C = c_j)}{\sum_{r=1}^{|C|} \Pr(C = c_r) \prod_{i=1}^{|A|} \Pr(A_i = a_i \mid C = c_r)}$$

# Classify a test instance

If we only need a decision on the most probable class for the test instance, we only need the numerator as its denominator is the same for every class. Thus, given a test example, we compute the following to decide the most probable class for the test instance

$$c = \arg \max_{c_j} \Pr(c_j) \prod_{i=1}^{|A|} \Pr(A_i = a_i \mid C = c_j)$$

# An example

- Compute all probabilities required for classification

| A | B | C |
|---|---|---|
| m | b | t |
| m | s | t |
| g | q | t |
| h | s | t |
| g | q | t |
| g | q | f |
| g | s | f |
| h | b | f |
| h | q | f |
| m | b | f |

$$\Pr(C = t) = 1/2,$$

$$\Pr(C = f) = 1/2$$

$$\Pr(A=m \mid C=t) = 2/5$$

$$\Pr(A=g \mid C=t) = 2/5$$

$$\Pr(A=h \mid C=t) = 1/5$$

$$\Pr(A=m \mid C=f) = 1/5$$

$$\Pr(A=g \mid C=f) = 2/5$$

$$\Pr(A=h \mid C=f) = 2/5$$

$$\Pr(B=b \mid C=t) = 1/5$$

$$\Pr(B=s \mid C=t) = 2/5$$

$$\Pr(B=q \mid C=t) = 2/5$$

$$\Pr(B=b \mid C=f) = 2/5$$

$$\Pr(B=s \mid C=f) = 1/5$$

$$\Pr(B=q \mid C=f) = 2/5$$

Now we have a test example:

$$A = m \quad B = q \quad C = ?$$

# An Example (cont ...)

For  $C = t$ , we have

$$\Pr(C = t) \prod_{j=1}^2 \Pr(A_j = a_j \mid C = t) = \frac{1}{2} \times \frac{2}{5} \times \frac{2}{5} = \frac{2}{25}$$

For class  $C = f$ , we have

$$\Pr(C = f) \prod_{j=1}^2 \Pr(A_j = a_j \mid C = f) = \frac{1}{2} \times \frac{1}{5} \times \frac{2}{5} = \frac{1}{25}$$

$C = t$  is more probable.  $t$  is the final class.

# Additional issues

**Numeric attributes:** Naïve Bayesian learning assumes that all attributes are categorical. Numeric attributes need to be discretized.

**Zero counts:** An particular attribute value never occurs together with a class in the training set. We need smoothing.

**Missing values:** Ignored

$$\Pr(A_i = a_i \mid C = c_j) = \frac{n_{ij} + \lambda}{n_j + \lambda n_i}$$

# On naïve Bayesian classifier

## **Advantages:**

Easy to implement

Very efficient

Good results obtained in many applications

## **Disadvantages**

Assumption: class conditional independence, therefore loss of accuracy when the assumption is seriously violated (those highly correlated data sets)

# Topic

Basic concepts

Decision tree induction

Evaluation of classifiers

Classification using association rules

Naïve Bayesian classification

**Naïve Bayes for text classification**

K-nearest neighbor

Ensemble methods: Bagging and Boosting

Summary

# Text classification/categorization

Due to the rapid growth of online documents in organizations and on the Web, automated document classification has become an important problem.

Techniques discussed previously can be applied to text classification, but they are not as effective as the next three methods.

We first study a naïve Bayesian method specifically formulated for texts, which makes use of some text specific features.

However, the ideas are similar to the preceding method.

# Probabilistic framework

**Generative model:** Each document is generated by a parametric distribution governed by a set of hidden parameters.

The generative model makes two assumptions

The data (or the text documents) are generated by a mixture model,

There is one-to-one correspondence between mixture components and document classes.

# Mixture model

A **mixture model** models the data with a number of statistical distributions.

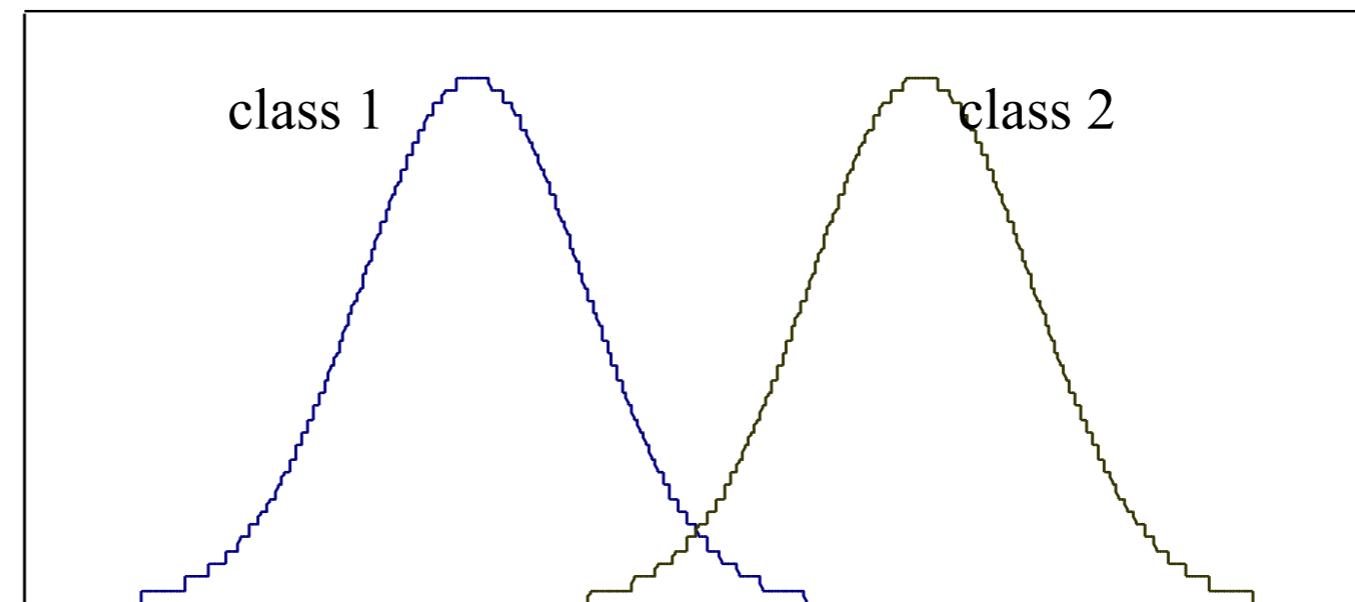
Intuitively, each distribution corresponds to a data cluster and the parameters of the distribution provide a description of the corresponding cluster.

Each distribution in a mixture model is also called a **mixture component**.

The distribution/component can be of any kind

# An example

The figure shows a plot of the **probability density function** of a 1-dimensional data set (with two classes) generated by a mixture of two Gaussian distributions, one per class, whose parameters (denoted by  $\theta_i$ ) are the mean ( $\mu_i$ ) and the standard deviation ( $\sigma_i$ ), i.e.,  $\theta_i = (\mu_i, \sigma_i)$ .



# Mixture model (cont ...)

Let the number of mixture components (or distributions) in a mixture model be  $K$ .

Let the  $j$ th distribution have the parameters  $\theta_j$ .

Let  $\Theta$  be the set of parameters of all components,  $\Theta = \{\varphi_1, \varphi_2, \dots, \varphi_K, \theta_1, \theta_2, \dots, \theta_K\}$ , where  $\varphi_j$  is the *mixture weight* (or *mixture probability*) of the mixture component  $j$  and  $\theta_j$  is the parameters of component  $j$ .

How does the model generate documents?

# Document generation

Due to one-to-one correspondence, each class corresponds to a mixture component.

The mixture weights are *class prior probabilities*, i.e.,  $\varphi_j = \Pr(c_j | \Theta)$ .

The mixture model generates each document  $d_i$  by:

first selecting a mixture component (or class) according to class prior probabilities (i.e., mixture weights),  $\varphi_j = \Pr(c_j | \Theta)$ .

then having this selected mixture component ( $c_j$ ) generate a document  $d_i$  according to its parameters, with distribution  $\Pr(d_i | c_j; \Theta)$  or more precisely  $\Pr(d_i | c_j; \theta_j)$ .

$$\Pr(d_i | \Theta) = \sum_{j=1}^{|C|} \Pr(c_j | \Theta) \Pr(d_i | c_j; \Theta) \quad (23)$$

# Model text documents

The naïve Bayesian classification treats each document as a “bag of words”. The generative model makes the following further assumptions:

Words of a document are generated independently of context given the class label. The familiar **naïve Bayes assumption** used before.

The probability of a word is **independent of its position** in the document.

The **document length** is chosen **independent of its class**.

# Multinomial distribution

With the assumptions, each document can be regarded as generated by a **multinomial distribution**.

In other words, each document is drawn from a multinomial distribution of words with as many independent trials as the length of the document.

The words are from a given vocabulary  $V = \{w_1, w_2, \dots, w_{|V|}\}$ .

## Use probability function of multinomial

$$\Pr(d_i | c_j; \Theta) = \Pr(|d_i|) |d_i|! \prod_{t=1}^{|V|} \frac{\Pr(w_t | c_j; \Theta)^{N_{ti}}}{N_{ti}!} \quad (24)$$

where  $N_{ti}$  is the number of times that word  $w_t$  occurs in document  $d_i$  and

$$\sum_{t=1}^{|V|} N_{it} = |d_i| \quad \sum_{t=1}^{|V|} \Pr(w_t | c_j; \Theta) = 1. \quad (25)$$

# Parameter estimation

The parameters are estimated based on empirical counts.

$$\Pr(w_t | c_j; \hat{\Theta}) = \frac{\sum_{i=1}^{|D|} N_{ti} \Pr(c_j | d_i)}{\sum_{s=1}^{|V|} \sum_{i=1}^{|D|} N_{si} \Pr(c_j | d_i)}. \quad (26)$$

In order to handle 0 counts for infrequent occurring words that do not appear in the training set, but may appear in the test set, we need to smooth the probability. *Lidstone smoothing*,  $0 \leq \lambda \leq 1$

$$\Pr(w_t | c_j; \hat{\Theta}) = \frac{\lambda + \sum_{i=1}^{|D|} N_{ti} \Pr(c_j | d_i)}{\lambda |V| + \sum_{s=1}^{|V|} \sum_{i=1}^{|D|} N_{si} \Pr(c_j | d_i)}. \quad (27)$$

# Parameter estimation (cont ...)

Class prior probabilities, which are mixture weights  $\varphi_j$ , can be easily estimated using training data

$$\Pr(c_j \mid \hat{\Theta}) = \frac{\sum_{i=1}^{|D|} \Pr(c_j \mid d_i)}{|D|} \quad (28)$$

# Classification

Given a test document  $d_i$ , from Eq. (23) (27) and (28)

$$\begin{aligned}\Pr(c_j | d_i; \hat{\Theta}) &= \frac{\Pr(c_j | \hat{\Theta}) \Pr(d_i | c_j; \hat{\Theta})}{\Pr(d_i | \hat{\Theta})} \\ &= \frac{\Pr(c_j | \hat{\Theta}) \prod_{k=1}^{|d_i|} \Pr(w_{d_i,k} | c_j; \hat{\Theta})}{\sum_{r=1}^{|C|} \Pr(c_r | \hat{\Theta}) \prod_{k=1}^{|d_i|} \Pr(w_{d_i,k} | c_r; \hat{\Theta})}\end{aligned}$$

where  $w_{d_i,k}$  is the word in position  $k$  of document  $d_i$ . If the final classifier is to classify each document into a single class, then the class with the highest posterior probability is selected:

$$\arg \max_{c_j \in C} \Pr(c_j | d_i; \hat{\Theta}) \quad (30)$$

# Discussions

Most assumptions made by naïve Bayesian learning are violated to some degree in practice.

Despite such violations, researchers have shown that naïve Bayesian learning produces very accurate models.

The main problem is the mixture model assumption. When this assumption is seriously violated, the classification performance can be poor.

Naïve Bayesian learning is extremely efficient.

# Topic

- Basic concepts
- Decision tree induction
- Evaluation of classifiers
- Classification using association rules
- Naïve Bayesian classification
- Naïve Bayes for text classification
- Support vector machines
- **K-nearest neighbor**
- Ensemble methods: Bagging and Boosting
- Summary

# k-Nearest Neighbor Classification (kNN)

- Unlike all the previous learning methods, kNN does not build model from the training data.
- To classify a test instance  $d$ , define  $k$ -neighborhood  $P$  as  $k$  nearest neighbors of  $d$
- Count number  $n$  of training instances in  $P$  that belong to class  $c_j$
- Estimate  $\Pr(c_j|d)$  as  $n/k$
- No training is needed. Classification time is linear in training set size for each test case.

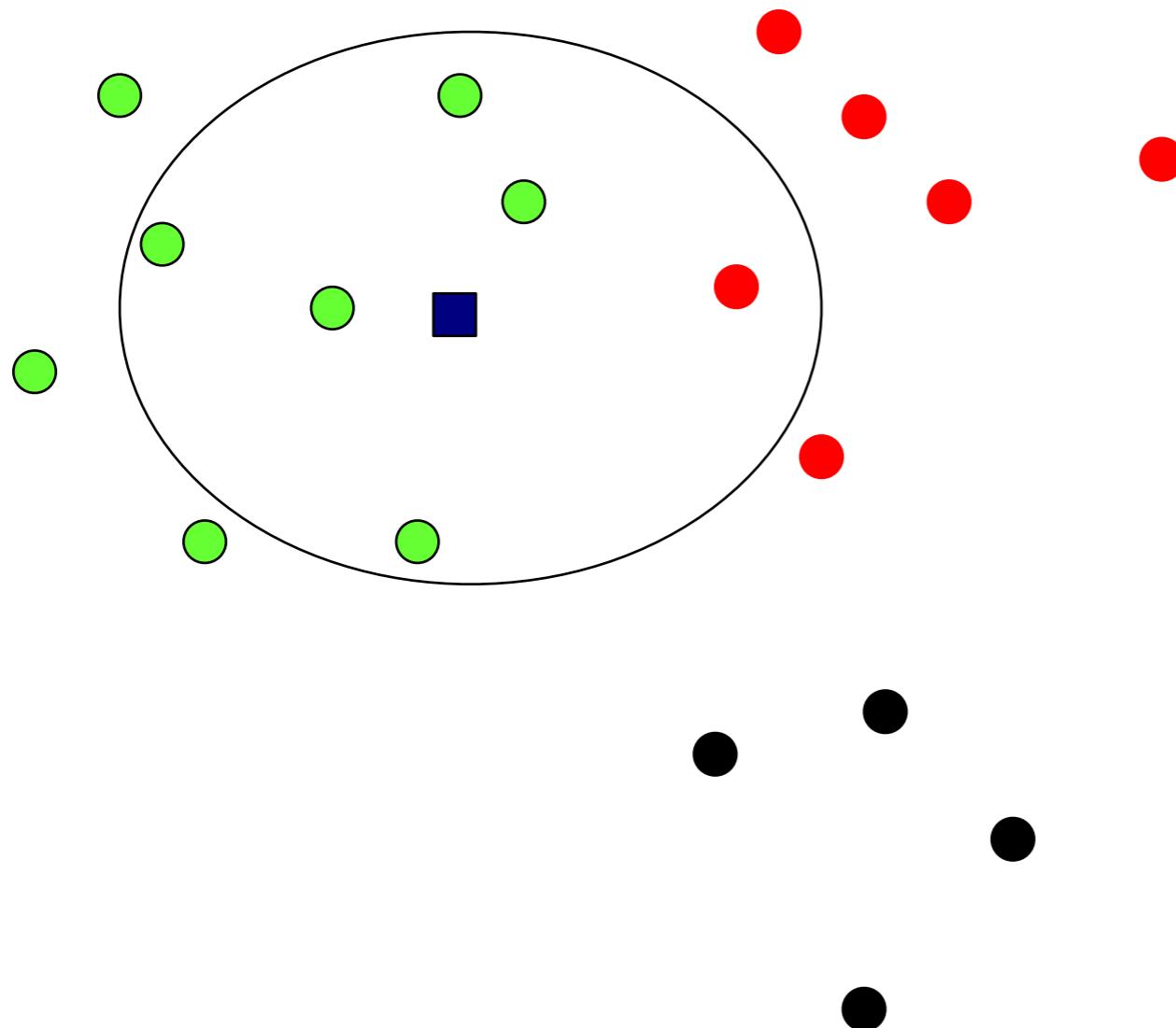
# kNNAlgorithm

**Algorithm**  $\text{kNN}(D, d, k)$

- 1 Compute the distance between  $d$  and every example in  $D$ ;
- 2 Choose the  $k$  examples in  $D$  that are nearest to  $d$ , denote the set by  $P$  ( $\subseteq D$ );
- 3 Assign  $d$  the class that is the most frequent class in  $P$  (or the majority class);

- $k$  is usually chosen empirically via a validation set or cross-validation by trying a range of  $k$  values.
- Distance function is crucial, but depends on applications.

# Example: k=6 (6NN)



- Government
- Science
- Arts

A new point ■  
Pr(science) ■?

# Discussions

- kNN can deal with complex and arbitrary decision boundaries.
- Despite its simplicity, researchers have shown that the classification accuracy of kNN can be quite strong and in many cases as accurate as those elaborated methods.
- kNN is slow at the classification time
- kNN does not produce an understandable model

# Topic

Basic concepts

Decision tree induction

Evaluation of classifiers

Classification using association rules

Naïve Bayesian classification

Naïve Bayes for text classification

K-nearest neighbor

**Ensemble methods: Bagging and Boosting**

Summary

# Combining classifiers

- So far, we have only discussed individual classifiers, i.e., how to build them and use them.
- **Can we combine multiple classifiers to produce a better classifier?**
- Yes, sometimes
- We discuss two main algorithms:
  - **Bagging**
  - **Boosting**

# Bagging

- Breiman, 1996
- Bootstrap Aggregating = Bagging
  - Application of **bootstrap sampling**
    - **Given:** set  $D$  containing  $m$  training examples
    - Create a sample  $S[i]$  of  $D$  by drawing  $m$  examples at random *with replacement* from  $D$
    - $S[i]$  of size  $m$ : expected to leave out 0.37 of examples from  $D$

# Bagging (cont...)

- **Training**
  - Create  $k$  bootstrap samples  $S[1], S[2], \dots, S[k]$
  - Build a distinct classifier on each  $S[i]$  to produce  $k$  classifiers, using the same learning algorithm.
- **Testing**
  - Classify each new instance by voting of the  $k$  classifiers (equal weights)

# Bagging Example

| Original       | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|----------------|---|---|---|---|---|---|---|---|
| Training set 1 | 2 | 7 | 8 | 3 | 7 | 6 | 3 | 1 |
| Training set 2 | 7 | 8 | 5 | 6 | 4 | 2 | 7 | 1 |
| Training set 3 | 3 | 6 | 2 | 7 | 5 | 6 | 2 | 2 |
| Training set 4 | 4 | 5 | 1 | 4 | 6 | 4 | 3 | 8 |

# Bagging (cont ...)

When does it help?

When learner is unstable

Small change to training set causes large change in the output classifier

True for decision trees, neural networks; not true for  $k$ -nearest neighbor, naïve Bayesian, class association rules

Experimentally, bagging can help substantially for unstable learners, may somewhat degrade results for stable learners

# Boosting

A family of methods:

We only study **AdaBoost** (Freund & Schapire, 1996)

## Training

Produce a sequence of classifiers (the same base learner)

Each classifier is dependent on the previous one, and focuses on the previous one's errors

Examples that are incorrectly predicted in previous classifiers are given higher weights

## Testing

For a test case, the results of the series of classifiers are combined to determine the final class of the test case.

# AdaBoost

## Weighted training set

$(x_1, y_1, w_1)$

$(x_2, y_2, w_2)$

...

$(x_n, y_n, w_n)$

Non-negative weights  
sum to 1



called a weaker classifier



- Build a classifier  $h_t$  whose accuracy on training set  $> \frac{1}{2}$  (better than random)

Change weights



# AdaBoost algorithm

## Algorithm AdaBoost.M1

**Input:** sequence of  $m$  examples  $\langle(x_1, y_1), \dots, (x_m, y_m)\rangle$

with labels  $y_i \in Y = \{1, \dots, k\}$

weak learning algorithm **WeakLearn**

integer  $T$  specifying number of iterations

**Initialize**  $D_1(i) = 1/m$  for all  $i$ .

**Do for**  $t = 1, 2, \dots, T$ :

1. Call **WeakLearn**, providing it with the distribution  $D_t$ .

2. Get back a hypothesis  $h_t : X \rightarrow Y$ .

3. Calculate the error of  $h_t$ :  $\epsilon_t = \sum_{i:h_t(x_i) \neq y_i} D_t(i)$ .

If  $\epsilon_t > 1/2$ , then set  $T = t - 1$  and abort loop.

4. Set  $\beta_t = \epsilon_t / (1 - \epsilon_t)$ .

5. Update distribution  $D_t$ :

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} \beta_t & \text{if } h_t(x_i) = y_i \\ 1 & \text{otherwise} \end{cases}$$

where  $Z_t$  is a normalization constant (chosen so that  $D_{t+1}$  will be a distribution).

**Output** the final hypothesis:

$$h_{fin}(x) = \arg \max_{y \in Y} \sum_{t: h_t(x) = y} \log \frac{1}{\beta_t}.$$

# Bagging, Boosting and C4.5

**C4.5's mean error rate over the 10 cross-validation.**

**Bagged C4.5**  
vs. C4.5.

**Boosted C4.5** vs. C4.5.

**Boosting vs. Bagging**

# Does AdaBoost always work?

The actual performance of boosting depends on the data and the base learner.

It requires the base learner to be unstable as bagging.

Boosting seems to be susceptible to noise.

When the number of outliers is very large, the emphasis placed on the hard examples can hurt the performance.

# Summary

- Applications of supervised learning are in almost any field or domain.
- We studied 8 classification techniques.
- There are still many other methods, e.g.,
  - Bayesian networks
  - Neural networks
  - Genetic algorithms
  - Fuzzy classification
- This large number of methods also show the importance of classification and its wide applicability.
- It remains to be an active research area.



# Unsupervised Learning

Fundamental Data Science for Data Scientist

# Supervised learning vs. unsupervised learning

**Supervised learning:** discover patterns in the data that relate data attributes with a target (class) attribute.

These patterns are then utilized to predict the values of the target attribute in future data instances.

**Unsupervised learning:** The data have no target attribute.

We want to explore the data to find some intrinsic structures in them.

# Clustering

- Clustering is a technique for finding **similarity groups** in data, called **clusters**. i.e.,
  - it groups data instances that are similar to (near) each other in one cluster and data instances that are very different (far away) from each other into different clusters.
- Clustering is often called an **unsupervised learning** task as no class values denoting an *a priori* grouping of the data instances are given, which is the case in supervised learning.
- Due to historical reasons, clustering is often considered synonymous with unsupervised learning.
  - In fact, association rule mining is also unsupervised
- This chapter focuses on clustering.

# An illustration

The data set has three natural groups of data points, i.e., 3 natural clusters.



# What is clustering for?

- Let us see some real-life examples
- **Example 1:** groups people of similar sizes together to make “small”, “medium” and “large” T-Shirts.
  - Tailor-made for each person: too expensive
  - One-size-fits-all: does not fit all.
- **Example 2:** In marketing, segment customers according to their similarities
  - To do targeted marketing.

# What is clustering for? (cont...)

**Example 3:** Given a collection of text documents, we want to organize them according to their content similarities,

To produce a topic hierarchy

**In fact, clustering is one of the most utilized data mining techniques.**

It has a long history, and used in almost every field, e.g., medicine, psychology, botany, sociology, biology, archeology, marketing, insurance, libraries, etc.

In recent years, due to the rapid increase of online documents, text clustering becomes important.

# Aspects of clustering

A clustering algorithm

Partitional clustering

Hierarchical clustering

...

A distance (similarity, or dissimilarity) function

Clustering quality

Inter-clusters distance  $\Rightarrow$  maximized

Intra-clusters distance  $\Rightarrow$  minimized

The **quality** of a clustering result depends on the algorithm, the distance function, and the application.

# K-means clustering

K-means is a **partitional clustering** algorithm

Let the set of data points (or instances)  $D$  be

$$\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\},$$

where  $\mathbf{x}_i = (x_{i1}, x_{i2}, \dots, x_{ir})$  is a **vector** in a real-valued space  $X \subseteq R^r$ ,

and  $r$  is the number of attributes (dimensions) in the data.

The  $k$ -means algorithm partitions the given data into  $k$  clusters.

Each cluster has a cluster **center**, called **centroid**.

$k$  is specified by the user

# K-means algorithm

Given  $k$ , the *k-means* algorithm works as follows:

- 1) Randomly choose  $k$  data points (**seeds**) to be the initial **centroids**, cluster centers
- 2) Assign each data point to the closest **centroid**
- 3) Re-compute the **centroids** using the current cluster memberships.
- 4) If a convergence criterion is not met, go to 2).

# K-means algorithm – (cont ...)

```
Algorithm k -means(k, D)
1 Choose k data points as the initial centroids (cluster centers)
2 repeat
3 for each data point $\mathbf{x} \in D$ do
4 compute the distance from \mathbf{x} to each centroid;
5 assign \mathbf{x} to the closest centroid // a centroid represents a cluster
6 endfor
7 re-compute the centroids using the current cluster memberships
8 until the stopping criterion is met
```

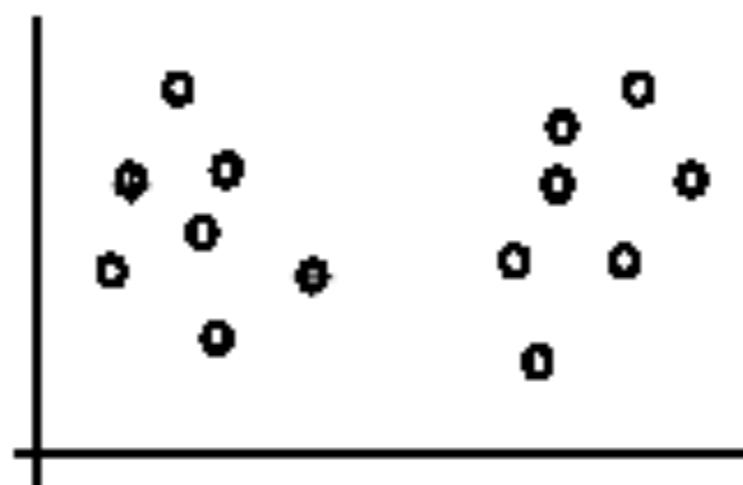
# Stopping/convergence criterion

1. no (or minimum) re-assignments of data points to different clusters,
2. no (or minimum) change of centroids, or
3. minimum decrease in the **sum of squared error** (SSE),

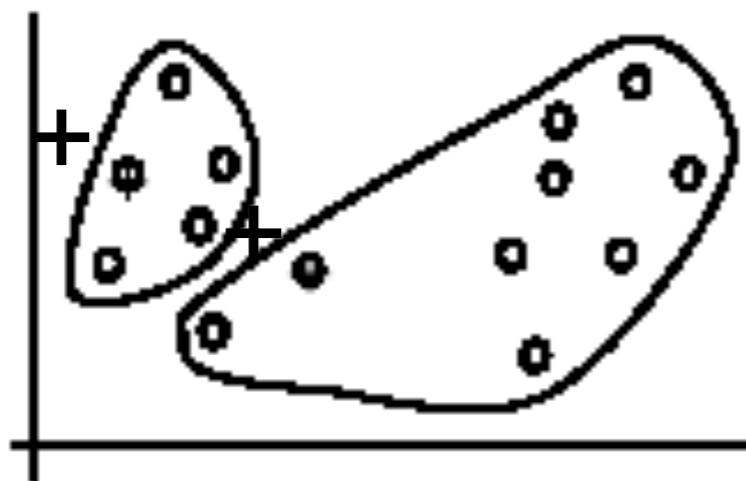
$C_j$  is the  $j$ th cluster,  $\mathbf{m}_j$  is the centroid of cluster  $C_j$  (the mean vector of all the data points in  $C_j$ ), and  $dist(\mathbf{x}, \mathbf{m}_j)$  is the distance between data point  $\mathbf{x}$  and centroid  $\mathbf{m}_j$ .

$$SSE = \sum_{j=1}^k \sum_{\mathbf{x} \in C_j} dist(\mathbf{x}, \mathbf{m}_j)^2 \quad (1)$$

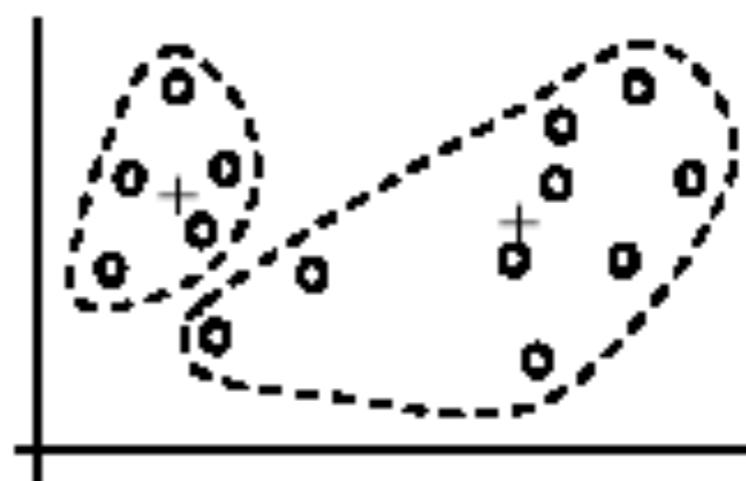
# An example



(A). Random selection of  $k$  centers

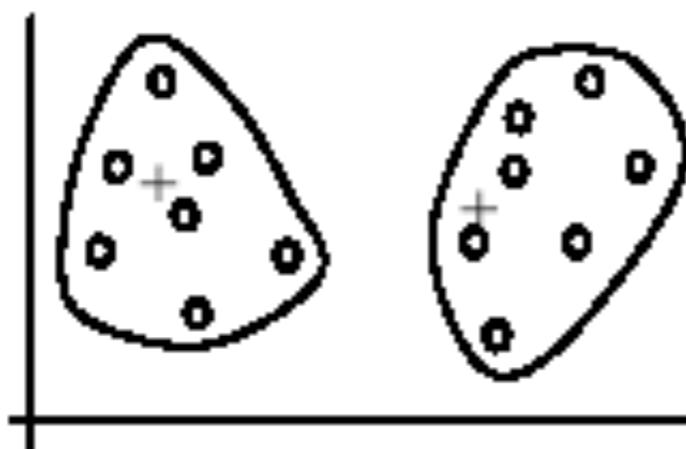


Iteration 1: (B). Cluster assignment

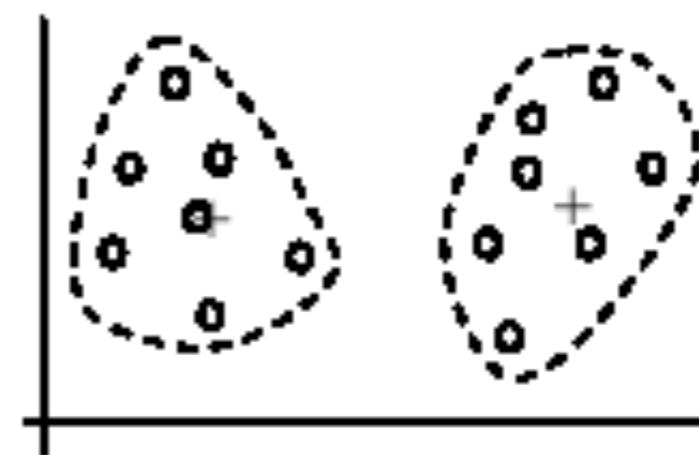


(C). Re-compute centroids

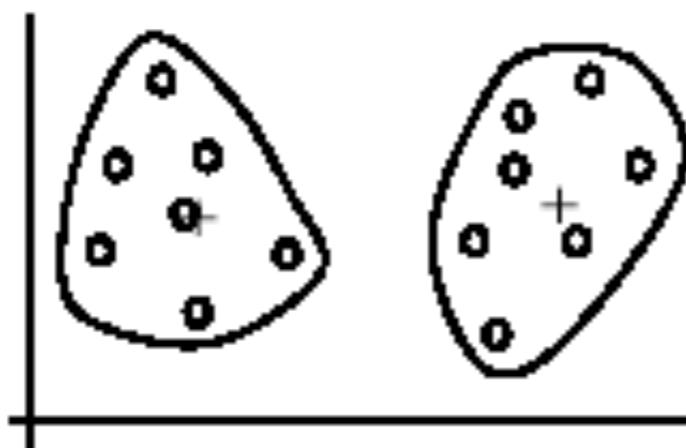
# An example (cont ...)



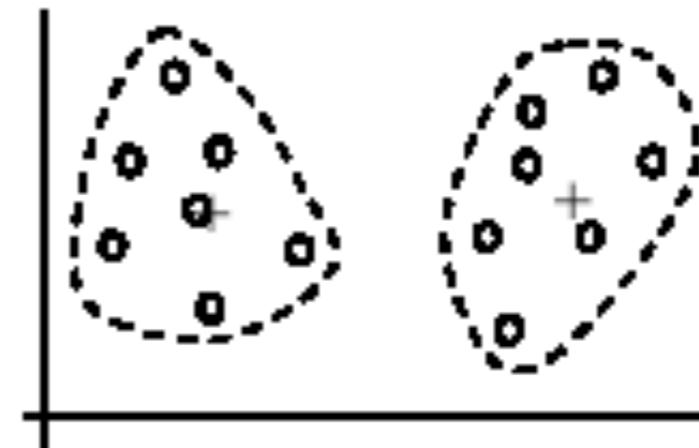
Iteration 2: (D). Cluster assignment



(E). Re-compute centroids



Iteration 3: (F). Cluster assignment



(G). Re-compute centroids

# An example distance function

The  $k$ -means algorithm can be used for any application data set where the **mean** can be defined and computed. In the **Euclidean space**, the mean of a cluster is computed with:

$$\mathbf{m}_j = \frac{1}{|C_j|} \sum_{\mathbf{x}_i \in C_j} \mathbf{x}_i \quad (2)$$

where  $|C_j|$  is the number of data points in cluster  $C_j$ . The distance from one data point  $\mathbf{x}_i$  to a mean (centroid)  $\mathbf{m}_j$  is computed with

$$\begin{aligned} dist(\mathbf{x}_i, \mathbf{m}_j) &= \| \mathbf{x}_i - \mathbf{m}_j \| \\ &= \sqrt{(x_{i1} - m_{j1})^2 + (x_{i2} - m_{j2})^2 + \dots + (x_{ir} - m_{jr})^2} \end{aligned} \quad (3)$$

# A disk version of $k$ -means

- K-means can be implemented with data on disk
  - In each iteration, it scans the data once.
  - as the centroids can be computed incrementally
- It can be used to cluster large datasets that do not fit in main memory
- We need to control the number of iterations
  - In practice, a limited is set (< 50).
- Not the best method. There are other scale-up algorithms, e.g., BIRCH (balanced iterative reducing and clustering using hierarchies).

## A disk version of k-means (cont ...)

**Algorithm** disk- $k$ -means( $k, D$ )

- 1 Choose  $k$  data points as the initial centroids  $\mathbf{m}_j, j = 1, \dots, k,$
- 2 **repeat**
- 3     initialize  $\mathbf{s}_j = \mathbf{0}, j = 1, \dots, k,$                                   //  $\mathbf{0}$  is a vector with all 0's
- 4     initialize  $n_j = 0, j = 1, \dots, k;$                                   //  $n_j$  is the number points in cluster  $j$
- 5     **for** each data point  $\mathbf{x} \in D$  **do**
- 6          $j = \arg \min_j \text{dist}(\mathbf{x}, \mathbf{m}_j);$
- 7         assign  $\mathbf{x}$  to the cluster  $j;$
- 8          $\mathbf{s}_j = \mathbf{s}_j + \mathbf{x};$
- 9          $n_j = n_j + 1;$
- 10      **endfor**
- 11      $\mathbf{m}_i = \mathbf{s}_i / n_i, i = 1, \dots, k,$
- 12   **until** the stopping criterion is met

# Strengths of k-means

## Strengths:

**Simple:** easy to understand and to implement

**Efficient:** Time complexity:  $O(tkn)$ ,

where  $n$  is the number of data points,

$k$  is the number of clusters, and

$t$  is the number of iterations.

Since both  $k$  and  $t$  are small. k-means is considered a linear algorithm.

K-means is the most popular clustering algorithm.

Note that: it terminates at a **local optimum** if SSE is used. The **global optimum** is hard to find due to complexity.

# Weaknesses of k-means

The algorithm is only applicable if the **mean** is defined.

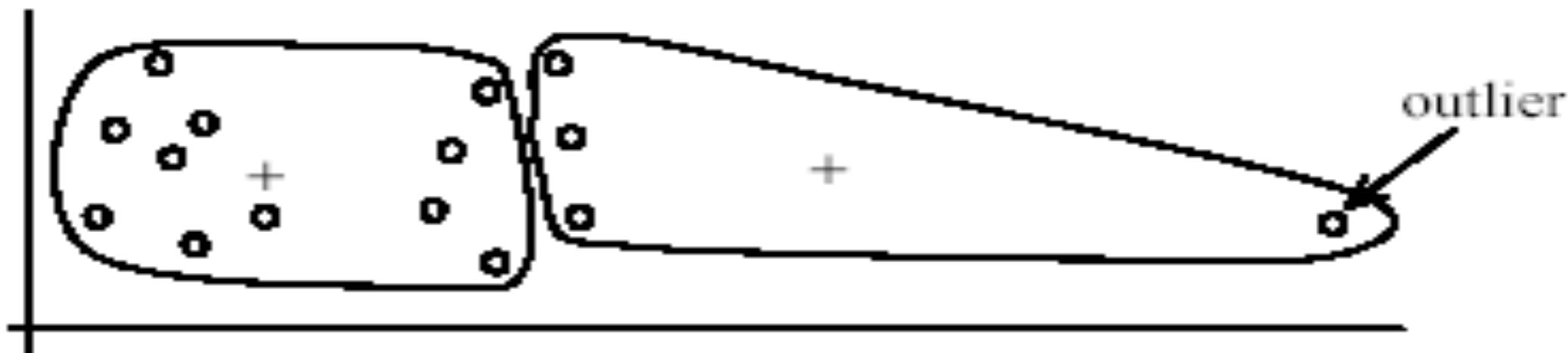
For categorical data, **k-mode** - the centroid is represented by most frequent values.

The user needs to specify ***k***.

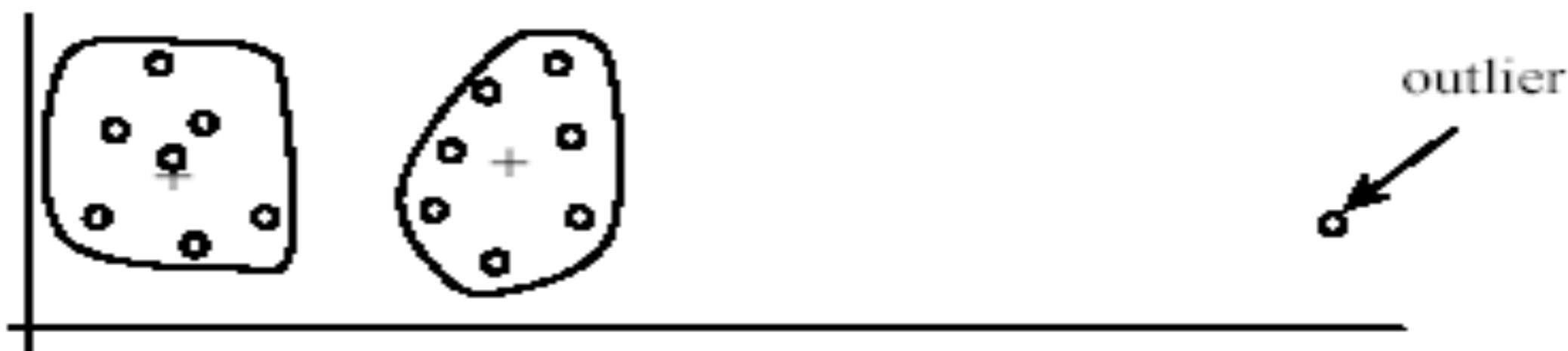
The algorithm is sensitive to **outliers**

- Outliers are data points that are very far away from other data points.
- Outliers could be errors in the data recording or some special data points with very different values.

## Weaknesses of k-means: Problems with outliers



(A): Undesirable clusters



(B): Ideal clusters

# Weaknesses of k-means: To deal with outliers

**One method** is to **remove some data points** in the clustering process that are much further away from the centroids than other data points.

To be safe, we may want to monitor these possible outliers over a few iterations and then decide to remove them.

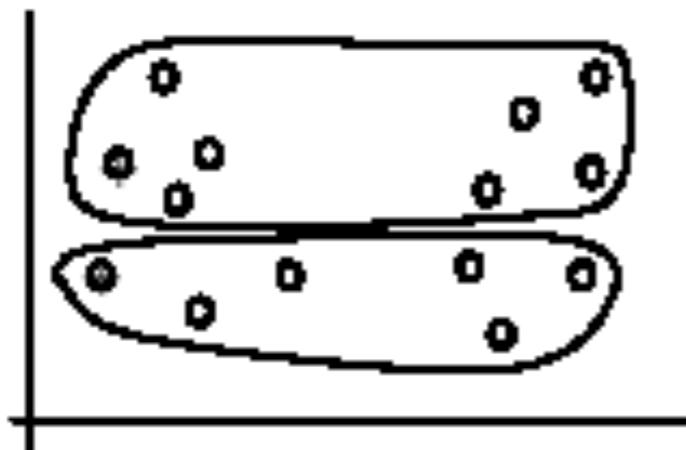
**Another method** is to **perform random sampling**. Since in sampling we only choose a small subset of the data points, the chance of selecting an outlier is very small.

- Assign the rest of the data points to the clusters by distance or similarity comparison, or classification

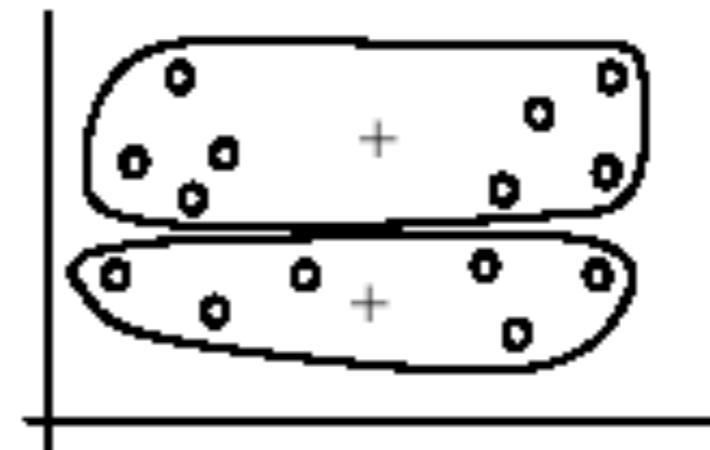
# Weaknesses of k-means (cont ...)



(A). Random selection of seeds (centroids)



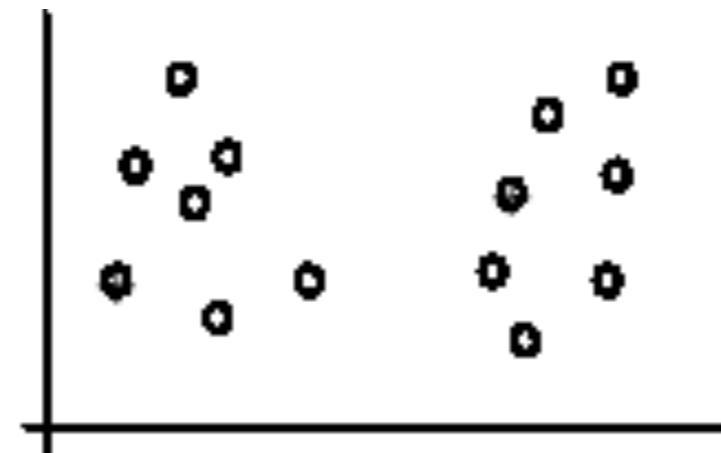
(B). Iteration 1



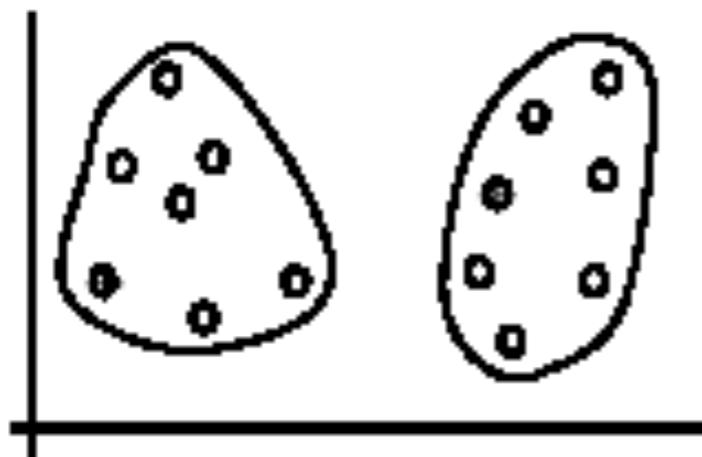
(C). Iteration 2

# Weaknesses of k-means (cont ...)

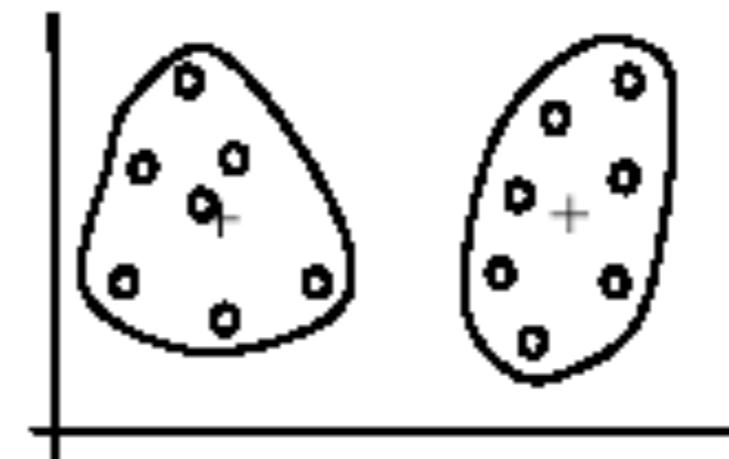
- There are some methods to help choose good seeds



(A). Random selection of  $k$  seeds (centroids)



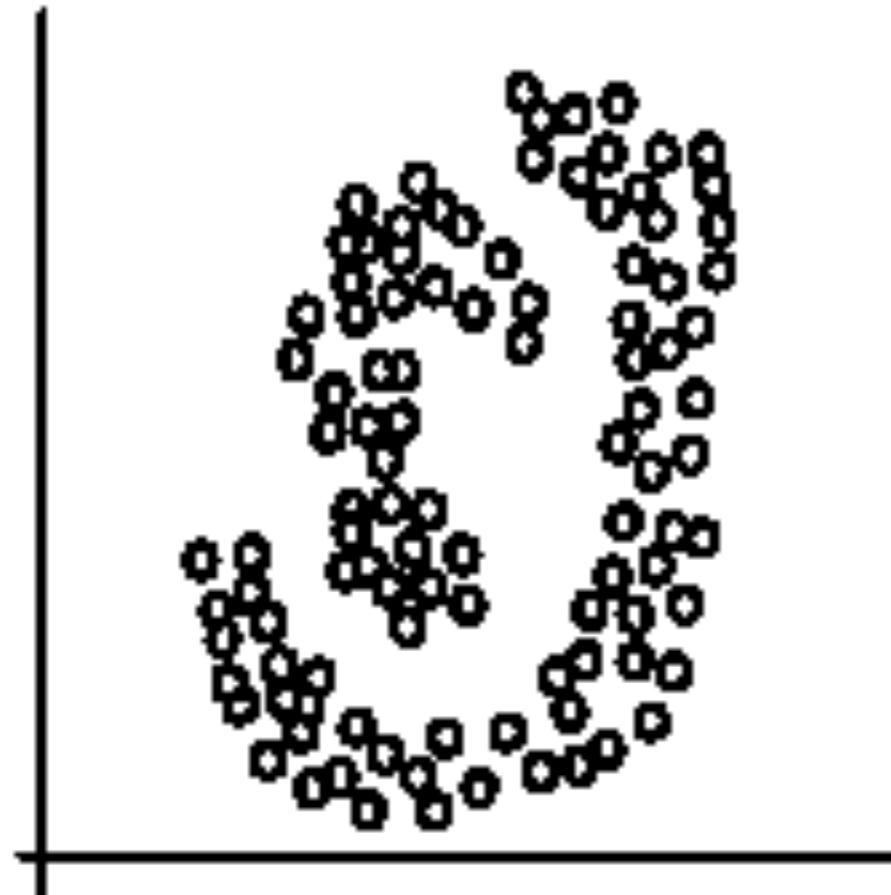
(B). Iteration 1



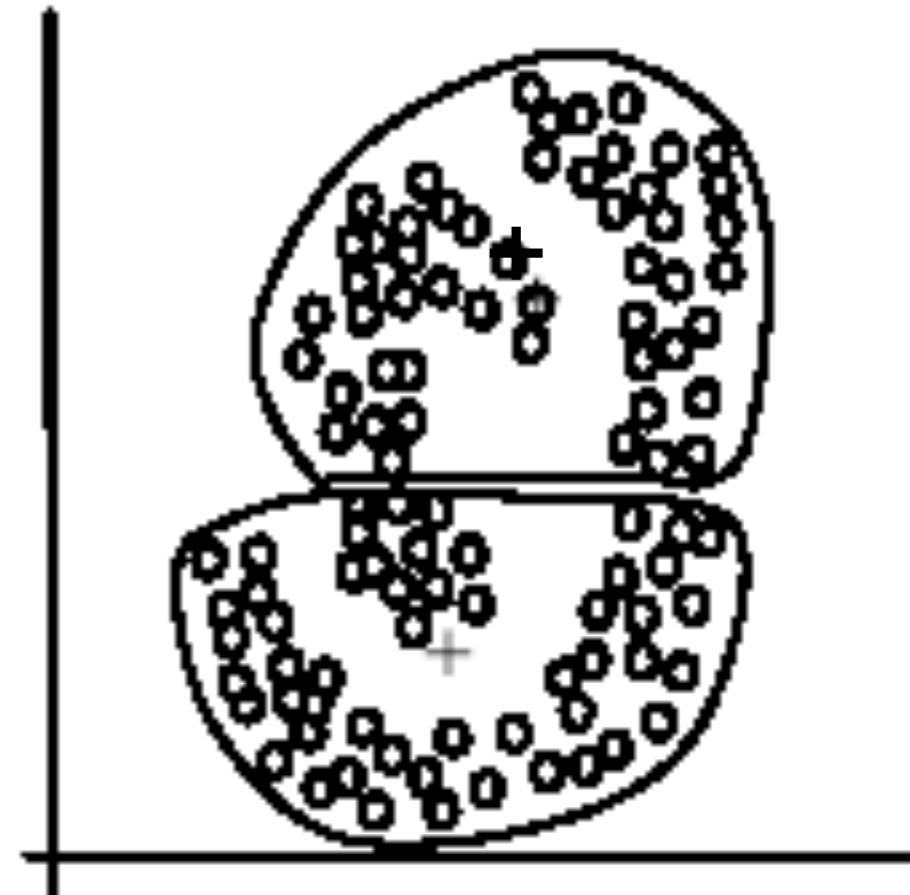
(C). Iteration 2

# Weaknesses of k-means (cont ...)

The  $k$ -means algorithm is not suitable for discovering clusters that are not hyper-ellipsoids (or hyper-spheres).



(A): Two natural clusters



(B):  $k$ -means clusters

# K-means summary

- Despite weaknesses,  $k$ -means is still the most popular algorithm due to its simplicity, efficiency and
  - other clustering algorithms have their own lists of weaknesses.
- No clear evidence that any other clustering algorithm performs better in general
  - although they may be more suitable for some specific types of data or applications.
- Comparing different clustering algorithms is a difficult task. No one knows the correct clusters!

# Common ways to represent clusters

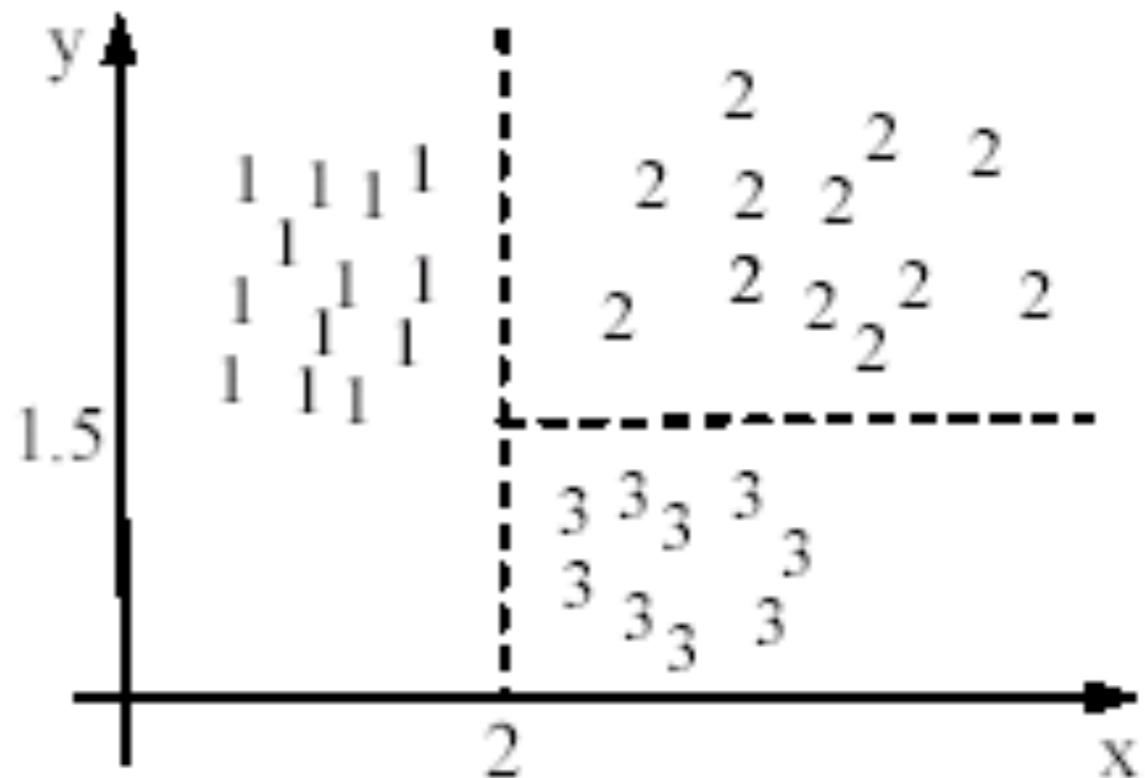
Use the centroid of each cluster to represent the cluster.

- compute the radius and
- standard deviation of the cluster to determine its spread in each dimension
- The centroid representation alone works well if the clusters are of the hyper-spherical shape.
- If clusters are elongated or are of other shapes, centroids are not sufficient

# Using classification model

All the data points in a cluster are regarded to have the same class label, e.g., the cluster ID.

run a supervised learning algorithm on the data to find a classification model.



$x \leq 2 \rightarrow$  cluster 1

$x > 2, y > 1.5 \rightarrow$  cluster 2

$x > 2, y \leq 1.5 \rightarrow$  cluster 3

# Use frequent values to represent cluster

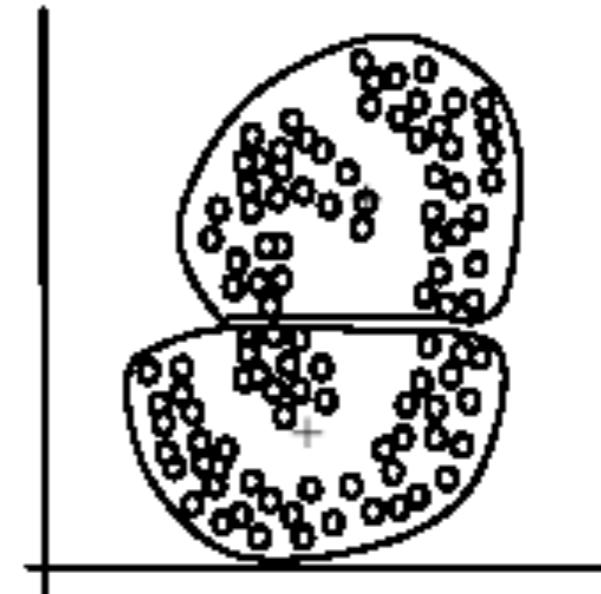
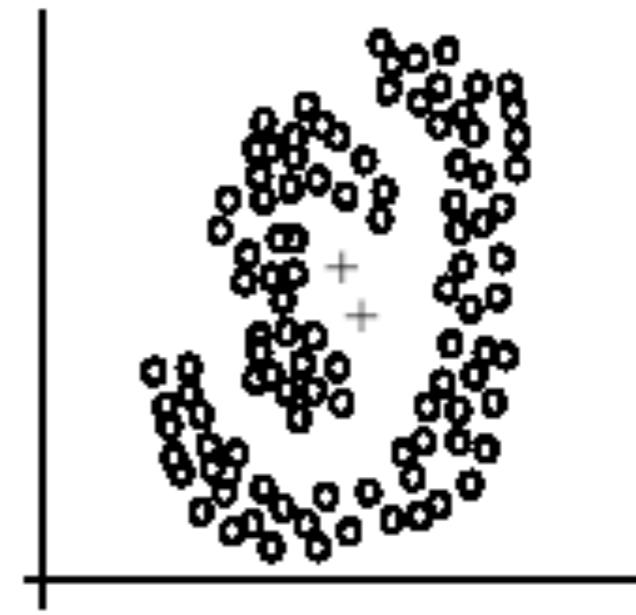
- This method is mainly for clustering of categorical data (e.g.,  $k$ -modes clustering).
- Main method used in text clustering, where a small set of frequent words in each cluster is selected to represent the cluster.

# Clusters of arbitrary shapes

Hyper-elliptical and hyper-spherical clusters are usually easy to represent, using their centroid together with spreads.

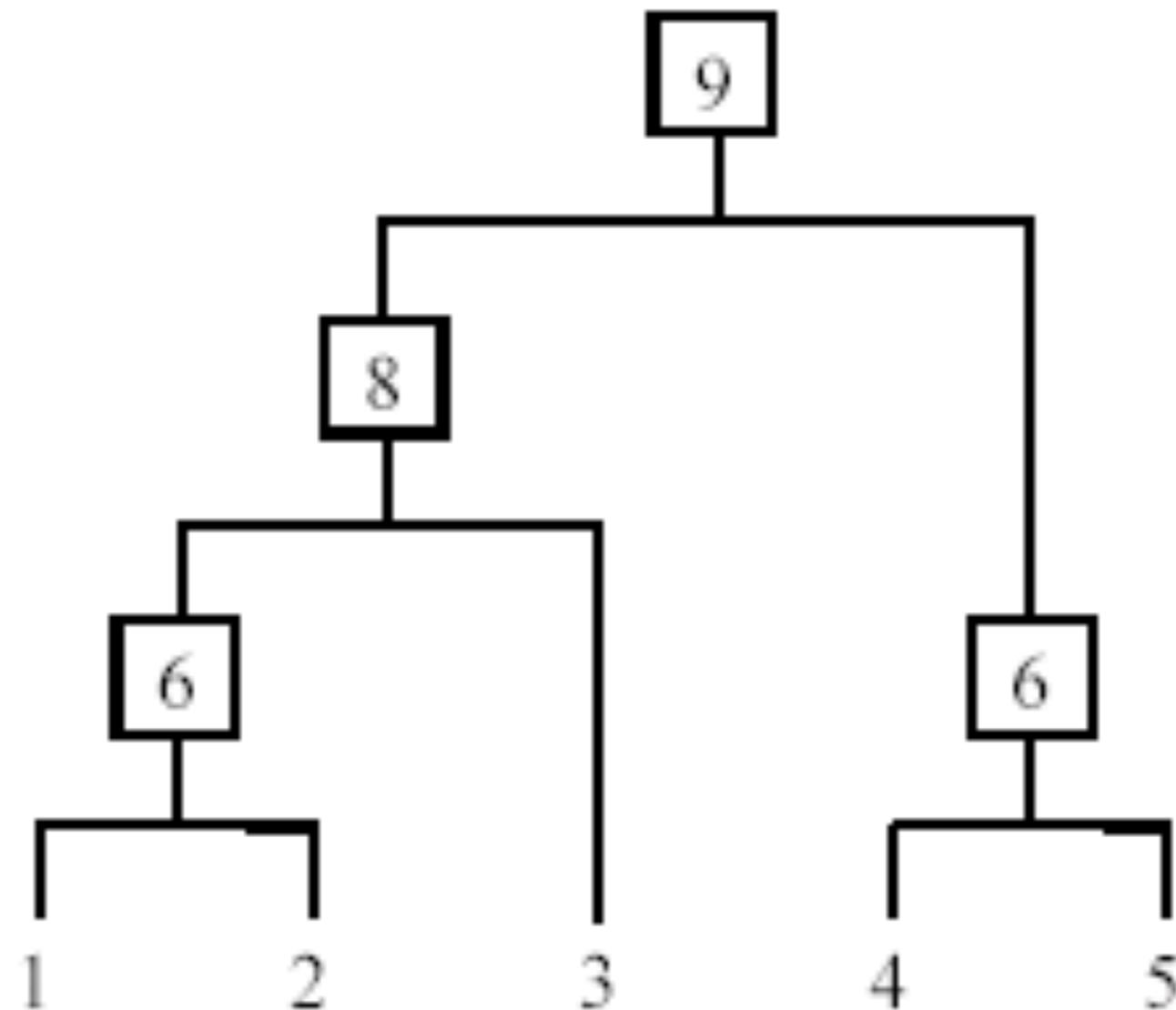
**Irregular shape clusters are hard to represent.** They may not be useful in some applications.

- Using centroids are not suitable (upper figure) in general
- K-means clusters may be more useful (lower figure), e.g., for making 2 size T-shirts.



# Hierarchical Clustering

Produce a nested sequence of clusters, a **tree**, also called **Dendrogram**.



# Types of hierarchical clustering

**Agglomerative (bottom up) clustering:** It builds the dendrogram (tree) from the bottom level, and

- merges the most similar (or nearest) pair of clusters
- stops when all the data points are merged into a single cluster (i.e., the root cluster).

**Divisive (top down) clustering:** It starts with all data points in one cluster, the root.

- Splits the root into a set of child clusters. Each child cluster is recursively divided further
- stops when only singleton clusters of individual data points remain, i.e., each cluster with only a single point

# Agglomerative clustering

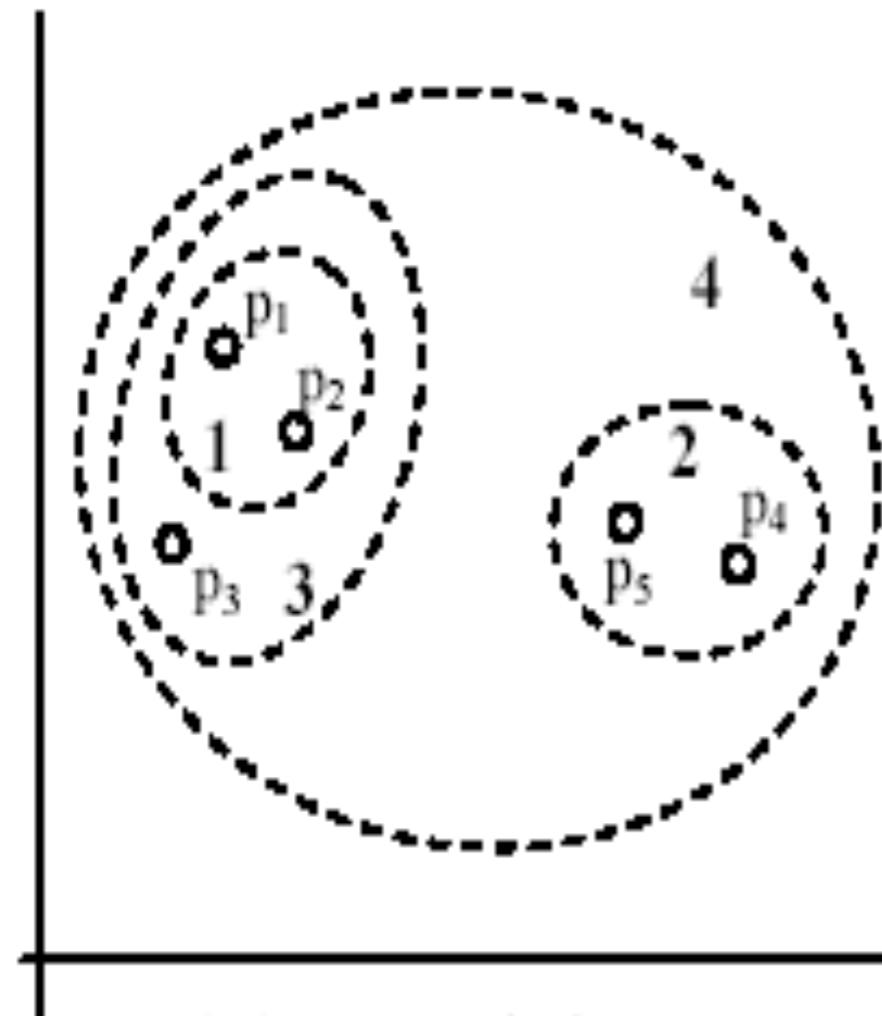
- It is more popular than divisive methods.
- At the beginning, each data point forms a cluster (also called a node).
- Merge nodes/clusters that have the least distance.
- Go on merging
- Eventually all nodes belong to one cluster

# Agglomerative clustering algorithm

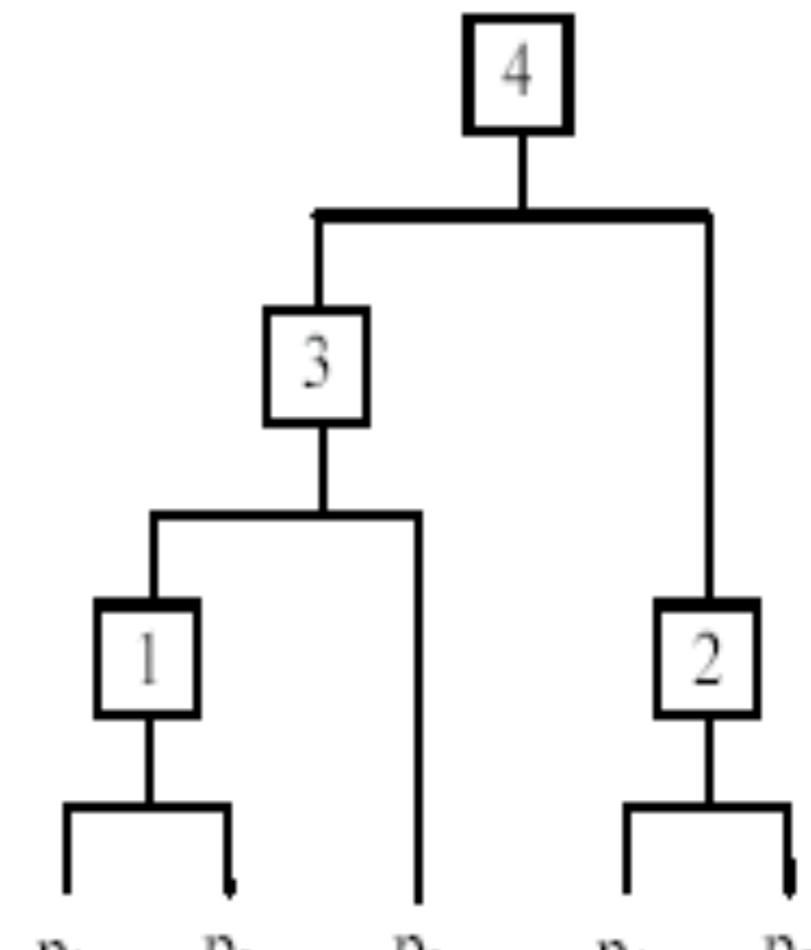
**Algorithm** Agglomerative( $D$ )

- 1 Make each data point in the data set  $D$  a cluster,
- 2 Compute all pair-wise distances of  $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n \in D$ ;
- 2 **repeat**
- 3     find two clusters that are nearest to each other;
- 4     merge the two clusters form a new cluster  $c$ ;
- 5     compute the distance from  $c$  to all other clusters;
- 12 **until** there is only one cluster left

# An example: working of the algorithm



(A). Nested clusters



(B) Dendrogram

# Measuring the distance of two clusters

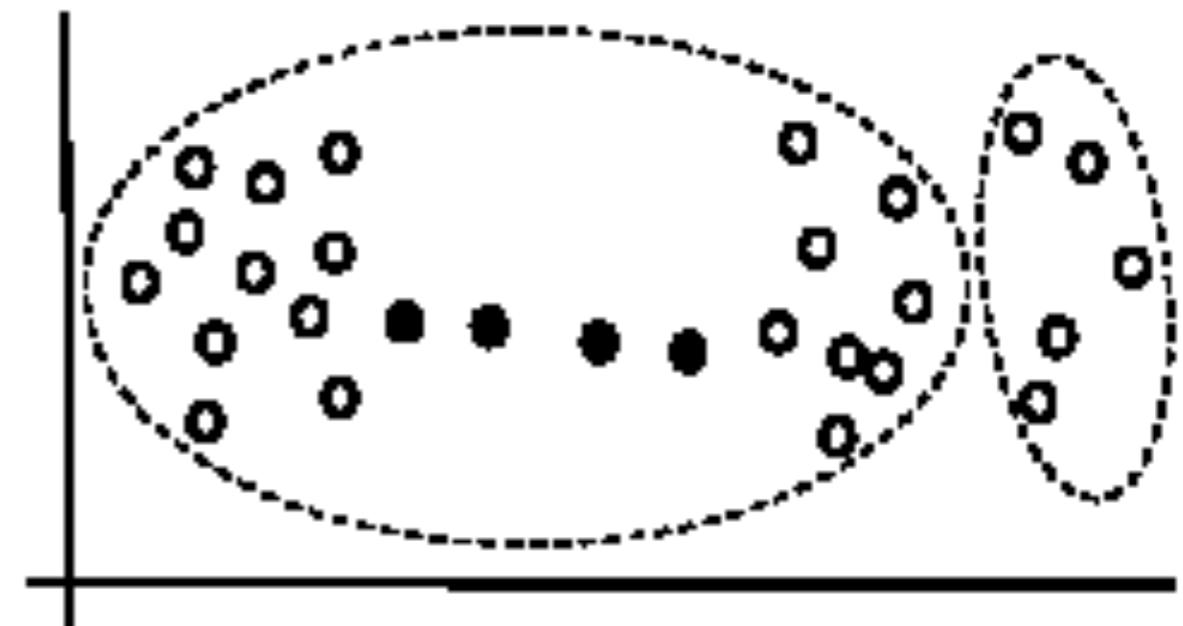
- A few ways to measure distances of two clusters.
- Results in different variations of the algorithm.
  - Single link
  - Complete link
  - Average link
  - Centroids
  - ...

# Single link method

The distance between two clusters is the distance between two **closest data points** in the two clusters, one data point from each cluster.

It can find arbitrarily shaped clusters, but

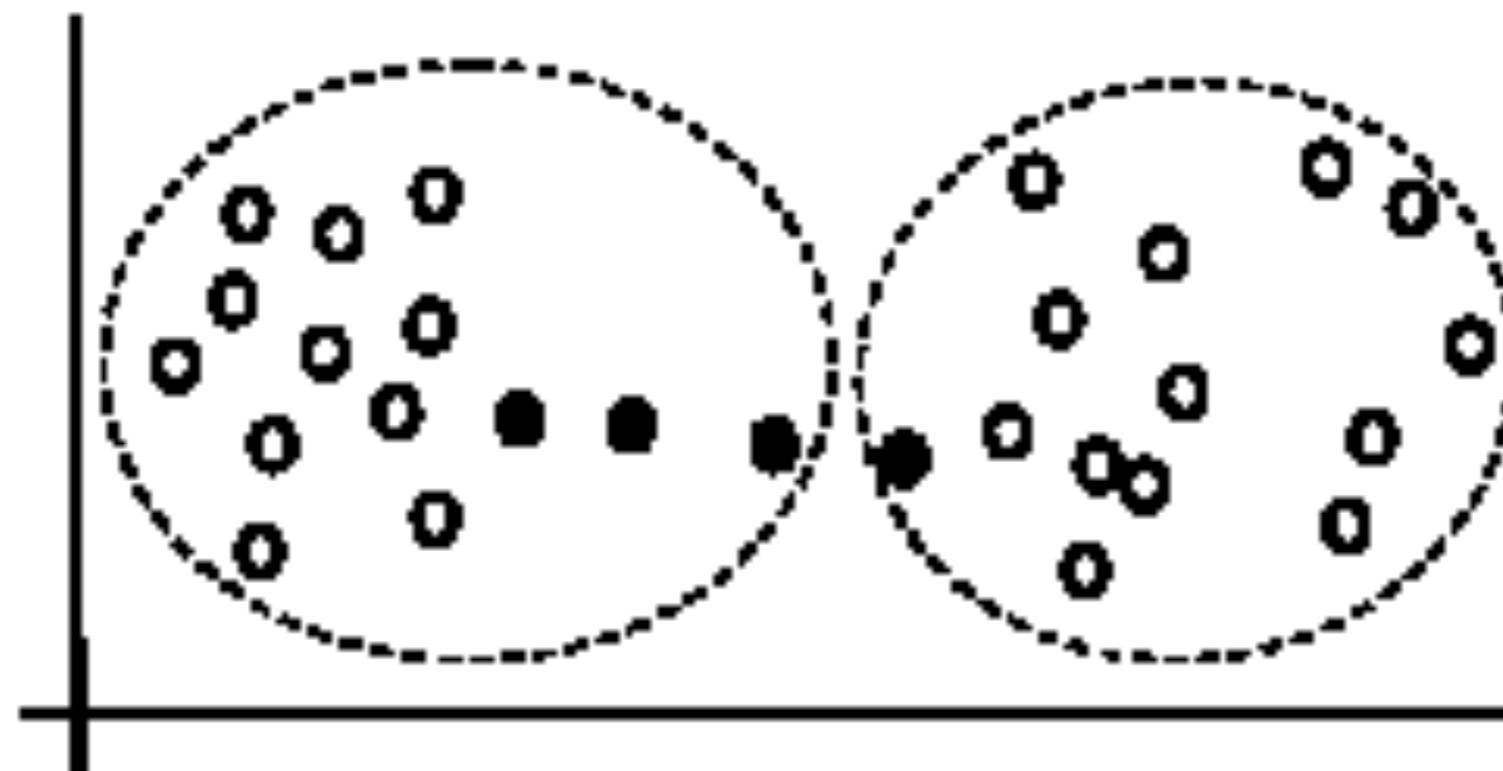
It may cause the undesirable “**chain effect**” by noisy points



Two natural clusters are split into two

# Complete link method

- The distance between two clusters is the distance of two **furthest** data points in the two clusters.
- It is sensitive to outliers because they are far away



# Average link and centroid methods

**Average link:** A compromise between

- the sensitivity of complete-link clustering to outliers and
- the tendency of single-link clustering to form long chains that do not correspond to the intuitive notion of clusters as compact, spherical objects.

In this method, the distance between two clusters is the average distance of all pair-wise distances between the data points in two clusters.

**Centroid method:** In this method, the distance between two clusters is the distance between their centroids

# The complexity

- All the algorithms are at least  $O(n^2)$ .  $n$  is the number of data points.
- Single link can be done in  $O(n^2)$ .
- Complete and average links can be done in  $O(n^2\log n)$ .
- Due to the complexity, hard to use for large data sets.
  - Sampling
  - Scale-up methods (e.g., BIRCH).

# Distance functions

- Key to clustering. “**similarity**” and “**dissimilarity**” can also commonly used terms.
- There are numerous distance functions for
  - Different types of data
    - Numeric data
    - Nominal data
  - Different specific applications

# Distance functions for numeric attributes

Most commonly used functions are

**Euclidean distance** and

**Manhattan (city block) distance**

We denote distance with:  $dist(\mathbf{x}_i, \mathbf{x}_j)$ , where  $\mathbf{x}_i$  and  $\mathbf{x}_j$  are data points (vectors)

They are special cases of **Minkowski distance**.  $h$  is positive integer.

$$dist(\mathbf{x}_i, \mathbf{x}_j) = \left( (x_{i1} - x_{j1})^h + (x_{i2} - x_{j2})^h + \dots + (x_{ir} - x_{jr})^h \right)^{\frac{1}{h}}$$

# Euclidean distance and Manhattan distance

If  $h = 2$ , it is the **Euclidean distance**

$$dist(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{(x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 + \dots + (x_{ir} - x_{jr})^2}$$

If  $h = 1$ , it is the **Manhattan distance**

$$dist(\mathbf{x}_i, \mathbf{x}_j) = |x_{i1} - x_{j1}| + |x_{i2} - x_{j2}| + \dots + |x_{ir} - x_{jr}|$$

**Weighted Euclidean distance**

$$dist(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{w_1(x_{i1} - x_{j1})^2 + w_2(x_{i2} - x_{j2})^2 + \dots + w_r(x_{ir} - x_{jr})^2}$$

# Squared distance and Chebychev distance

**Squared Euclidean distance:** to place progressively greater weight on data points that are further apart.

$$dist(\mathbf{x}_i, \mathbf{x}_j) = (x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 + \dots + (x_{ir} - x_{jr})^2$$

**Chebychev distance:** one wants to define two data points as "different" if they are different on any one of the attributes.

$$dist(\mathbf{x}_i, \mathbf{x}_j) = \max(|x_{i1} - x_{j1}|, |x_{i2} - x_{j2}|, \dots, |x_{ir} - x_{jr}|)$$

# Distance functions for binary and nominal attributes

**Binary attribute:** has two values or states but no ordering relationships,  
e.g.,

Gender: male and female.

We use a confusion matrix to introduce the distance functions/measures.

Let the  $i^{\text{th}}$  and  $j^{\text{th}}$  data points be  $\mathbf{x}_i$  and  $\mathbf{x}_j$  (vectors)

# Confusion matrix

|                |   | Data point $j$ |       | (10)      |
|----------------|---|----------------|-------|-----------|
|                |   | 1              | 0     |           |
| Data point $i$ | 1 | $a$            | $b$   | $a+b$     |
|                | 0 | $c$            | $d$   | $c+d$     |
|                |   | $a+c$          | $b+d$ | $a+b+c+d$ |

- $a$ : the number of attributes with the value of 1 for both data points.
- $b$ : the number of attributes for which  $x_{if}=1$  and  $x_{jf}=0$ , where  $x_{if}$  ( $x_{jf}$ ) is the value of the  $f$ th attribute of the data point  $\mathbf{x}_i$  ( $\mathbf{x}_j$ ).
- $c$ : the number of attributes for which  $x_{if}=0$  and  $x_{jf}=1$ .
- $d$ : the number of attributes with the value of 0 for both data points.

# Symmetric binary attributes

A binary attribute is **symmetric** if both of its states (0 and 1) have equal importance, and carry the same weights, e.g., male and female of the attribute Gender

Distance function: [Simple Matching Coefficient](#), proportion of mismatches of their values

$$dist(\mathbf{x}_i, \mathbf{x}_j) = \frac{b + c}{a + b + c + d}$$

# Symmetric binary attributes: example

|                |   |   |   |   |   |   |   |
|----------------|---|---|---|---|---|---|---|
| $\mathbf{x}_1$ | 1 | 1 | 1 | 0 | 1 | 0 | 0 |
| $\mathbf{x}_2$ | 0 | 1 | 1 | 0 | 0 | 1 | 0 |

$$dist(\mathbf{x}_i, \mathbf{x}_j) = \frac{2+1}{2+2+1+2} = \frac{3}{7} = 0.429$$

# Asymmetric binary attributes

**Asymmetric:** if one of the states is more important or more valuable than the other.

By convention, state 1 represents the more important state, which is typically the rare or infrequent state.

**Jaccard coefficient** is a popular measure

$$dist(\mathbf{x}_i, \mathbf{x}_j) = \frac{b + c}{a + b + c}$$

We can have some variations, adding weights

# Nominal attributes

**Nominal attributes:** with more than two states or values.

the commonly used distance measure is also based on the [simple matching method](#).

Given two data points  $\mathbf{x}_i$  and  $\mathbf{x}_j$ , let the number of attributes be  $r$ , and the number of values that match in  $\mathbf{x}_i$  and  $\mathbf{x}_j$  be  $q$ .

$$dist(\mathbf{x}_i, \mathbf{x}_j) = \frac{r - q}{r}$$

# Distance function for text documents

- A text document consists of a sequence of sentences and each sentence consists of a sequence of words.
- To simplify: a document is usually considered a “bag” of words in document clustering.
  - Sequence and position of words are ignored.
- A document is represented with a vector just like a normal data point.
- It is common to use similarity to compare two documents rather than distance.
  - The most commonly used similarity function is the **cosine similarity**.

# Data standardization

- In the Euclidean space, standardization of attributes is recommended so that all attributes can have equal impact on the computation of distances.
- Consider the following pair of data points
  - $\mathbf{x}_i: (0.1, 20)$  and  $\mathbf{x}_j: (0.9, 720)$ .

$$dist(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{(0.9 - 0.1)^2 + (720 - 20)^2} = 700.000457,$$

- The distance is almost completely dominated by  $(720-20) = 700$ .
- **Standardize attributes:** to force the attributes to have a common value range

# Interval-scaled attributes

- Their values are real numbers following a linear scale.
  - The difference in Age between 10 and 20 is the same as that between 40 and 50.
  - The key idea is that intervals keep the same importance through out the scale
- Two main approaches to standardize interval scaled attributes, **range** and **z-score**.  $f$  is an attribute

$$range(x_{if}) = \frac{x_{if} - \min(f)}{\max(f) - \min(f)},$$

# Interval-scaled attributes (cont ...)

**Z-score**: transforms the attribute values so that they have a mean of zero and a **mean absolute deviation** of 1. The mean absolute deviation of attribute  $f$ , denoted by  $s_f$ , is computed as follows

$$s_f = \frac{1}{n} (|x_{1f} - m_f| + |x_{2f} - m_f| + \dots + |x_{nf} - m_f|)$$

$$m_f = \frac{1}{n} (x_{1f} + x_{2f} + \dots + x_{nf})$$

Z-score: 
$$z(x_{if}) = \frac{x_{if} - m_f}{s_f}.$$

# Ratio-scaled attributes

Numeric attributes, but unlike interval-scaled attributes, their scales are exponential,

For example, the total amount of microorganisms that evolve in a time  $t$  is approximately given by

$$Ae^{Bt},$$

where  $A$  and  $B$  are some positive constants.

Do log transform:

$$\log(x_{if})$$

Then treat it as an interval-scaled attribute

# Nominal attributes

- Sometime, we need to transform nominal attributes to numeric attributes.
- Transform nominal attributes to binary attributes.
  - The number of values of a nominal attribute is  $v$ .
  - Create  $v$  binary attributes to represent them.
  - If a data instance for the nominal attribute takes a particular value, the value of its binary attribute is set to 1, otherwise it is set to 0.
- The resulting binary attributes can be used as numeric attributes, with two values, 0 and 1.

# Nominal attributes: an example

- Nominal attribute *fruit*: has three values,
  - Apple, Orange, and Pear
- We create three binary attributes called, Apple, Orange, and Pear in the new data.
- If a particular data instance in the original data has Apple as the value for *fruit*,
  - then in the transformed data, we set the value of the attribute Apple to 1, and
  - the values of attributes Orange and Pear to 0

# Ordinal attributes

- Ordinal attribute: an ordinal attribute is like a nominal attribute, but its values have a numerical ordering. E.g.,
  - Age attribute with values: Young, MiddleAge and Old. They are ordered.
  - Common approach to standardization: treat as an interval-scaled attribute.

# Mixed attributes

- Our distance functions given are for data with all numeric attributes, or all nominal attributes, etc.
- Practical data has different types:
  - Any subset of the 6 types of attributes,
  - **interval-scaled**,
  - **symmetric binary**,
  - **asymmetric binary**,
  - **ratio-scaled**,
  - **ordinal** and
  - **nominal**

# Convert to a single type

- One common way of dealing with mixed attributes is to
  - Decide the dominant attribute type, and
  - Convert the other types to this type.
- E.g, if most attributes in a data set are interval-scaled,
  - we convert ordinal attributes and ratio-scaled attributes to interval-scaled attributes.
  - It is also appropriate to treat symmetric binary attributes as interval-scaled attributes.

# Convert to a single type (cont ...)

- It does not make much sense to convert a **nominal attribute** or an **asymmetric binary** attribute to an interval-scaled attribute,
  - but it is still frequently done in practice by assigning some numbers to them according to some hidden ordering, e.g., prices of the fruits
- Alternatively, a nominal attribute can be converted to a set of (symmetric) binary attributes, which are then treated as numeric attributes.

# Combining individual distances

This approach computes individual attribute distances and then combine them.

$$dist(\mathbf{x}_i, \mathbf{x}_j) = \frac{\sum_{f=1}^r \delta_{ij}^f d_{ij}^f}{\sum_{f=1}^r \delta_{ij}^f}$$

This distance value is between 0 and 1.  $r$  is the number of attributes in the data set. The indicator  $\delta_{ij}^f$  is 1 when both values  $x_{if}$  and  $x_{jf}$  for attribute  $f$  are non-missing, and it is set to 0 otherwise. It is also set to 0 if attribute  $f$  is asymmetric and the match is 0-0. Equation (25) cannot be computed if all  $\delta_{ij}^f$ 's are 0. In such a case, some default value may be used or one of the data points is removed.

$d_{ij}^f$  is the distance contributed by attribute  $f$ , and it is in the 0-1 range.

# How to choose a clustering algorithm

- Clustering research has a long history. A vast collection of algorithms are available.
  - We only introduced several main algorithms.
- Choosing the “best” algorithm is a challenge.
  - Every algorithm has limitations and works well with certain data distributions.
  - It is very hard, if not impossible, to know what distribution the application data follow. The data may not fully follow any “ideal” structure or distribution required by the algorithms.
  - One also needs to decide how to standardize the data, to choose a suitable distance function and to select other parameter values.

# Choose a clustering algorithm (cont ...)

- Due to these complexities, the common practice is to
  - run several algorithms using different distance functions and parameter settings, and
  - then carefully analyze and compare the results.
- The interpretation of the results must be based on insight into the meaning of the original data together with knowledge of the algorithms used.
- Clustering is highly **application dependent** and to certain extent **subjective** (personal preferences).

# Cluster Evaluation: hard problem

The quality of a clustering is very hard to evaluate because

We do not know the correct clusters

Some methods are used:

User inspection

- Study centroids, and spreads
- Rules from a decision tree.
- For text documents, one can read some documents in clusters.

# Cluster evaluation: ground truth

We use some labeled data (for classification)

**Assumption:** Each class is a cluster.

After clustering, a confusion matrix is constructed. From the matrix, we compute various measurements, **entropy**, **purity**, **precision**, **recall** and **F-score**.

Let the classes in the data  $D$  be  $C = (c_1, c_2, \dots, c_k)$ .

The clustering method produces  $k$  clusters, which divides  $D$  into  $k$  disjoint subsets,  $D_1, D_2, \dots, D_k$ .

# Evaluation measures: Entropy

**Entropy:** For each cluster, we can measure its entropy as follows:

$$\text{entropy}(D_i) = -\sum_{j=1}^k \Pr_i(c_j) \log_2 \Pr_i(c_j), \quad (29)$$

where  $\Pr_i(c_j)$  is the proportion of class  $c_j$  data points in cluster  $i$  or  $D_i$ . The total entropy of the whole clustering (which considers all clusters) is

$$\text{entropy}_{total}(D) = \sum_{i=1}^k \frac{|D_i|}{|D|} \times \text{entropy}(D_i) \quad (30)$$

# Evaluation measures: purity

**Purity:** This again measures the extent that a cluster contains only one class of data. The purity of each cluster is computed with

$$purity(D_i) = \max_j(\Pr_i(c_j)) \quad (31)$$

The total purity of the whole clustering (considering all clusters) is

$$purity_{total}(D) = \sum_{i=1}^k \frac{|D_i|}{|D|} \times purity(D_i) \quad (32)$$

## An example

**Example 14:** Assume we have a text collection  $D$  of 900 documents from three topics (or three classes), Science, Sports, and Politics. Each class has 300 documents. Each document in  $D$  is labeled with one of the topics (classes). We use this collection to perform clustering to find three clusters. Note that class/topic labels are not used in clustering. After clustering, we want to measure the effectiveness of the clustering algorithm.

| Cluster | Science | Sports | Politics |  | Entropy | Purity |
|---------|---------|--------|----------|--|---------|--------|
| 1       | 250     | 20     | 10       |  | 0.589   | 0.893  |
| 2       | 20      | 180    | 80       |  | 1.198   | 0.643  |
| 3       | 30      | 100    | 210      |  | 1.257   | 0.617  |
| Total   | 300     | 300    | 300      |  | 1.031   | 0.711  |

# A remark about ground truth evaluation

Commonly used to compare different clustering algorithms.

A real-life data set for clustering has no class labels.

Thus although an algorithm may perform very well on some labeled data sets, no guarantee that it will perform well on the actual application data at hand.

The fact that it performs well on some label data sets does give us some confidence of the quality of the algorithm.

This evaluation method is said to be based on **external data** or information.

# Evaluation based on internal information

## **Intra-cluster cohesion** (compactness):

Cohesion measures how near the data points in a cluster are to the cluster centroid.

Sum of squared error (SSE) is a commonly used measure.

## **Inter-cluster separation** (isolation):

Separation means that different cluster centroids should be far away from one another.

In most applications, expert judgments are still the key.

# Indirect evaluation

- In some applications, clustering is **not the primary task**, but used to help perform another task.
- We can use the performance on the primary task to compare clustering methods.
- For instance, in an application, the primary task is to provide recommendations on book purchasing to online shoppers.
  - If we can cluster books according to their features, we might be able to provide better recommendations.
  - We can evaluate different clustering algorithms based on how well they help with the recommendation task.
  - Here, we assume that the recommendation can be reliably evaluated.

# Holes in data space

- All the clustering algorithms only group data.
- Clusters only represent one aspect of the knowledge in the data.
- Another aspect that we have not studied is the **holes**.

A hole is a region in the data space that contains no or few data points. Reasons:

- insufficient data in certain areas, and/or certain attribute-value combinations are not possible or seldom occur.

# Holes are useful too

Although clusters are important, holes in the space can be quite useful too.

For example, in a disease database

we may find that certain symptoms and/or test values do not occur together, or

when a certain medicine is used, some test values never go beyond certain ranges.

Discovery of such information can be important in medical domains because

it could mean the discovery of a cure to a disease or some biological laws.

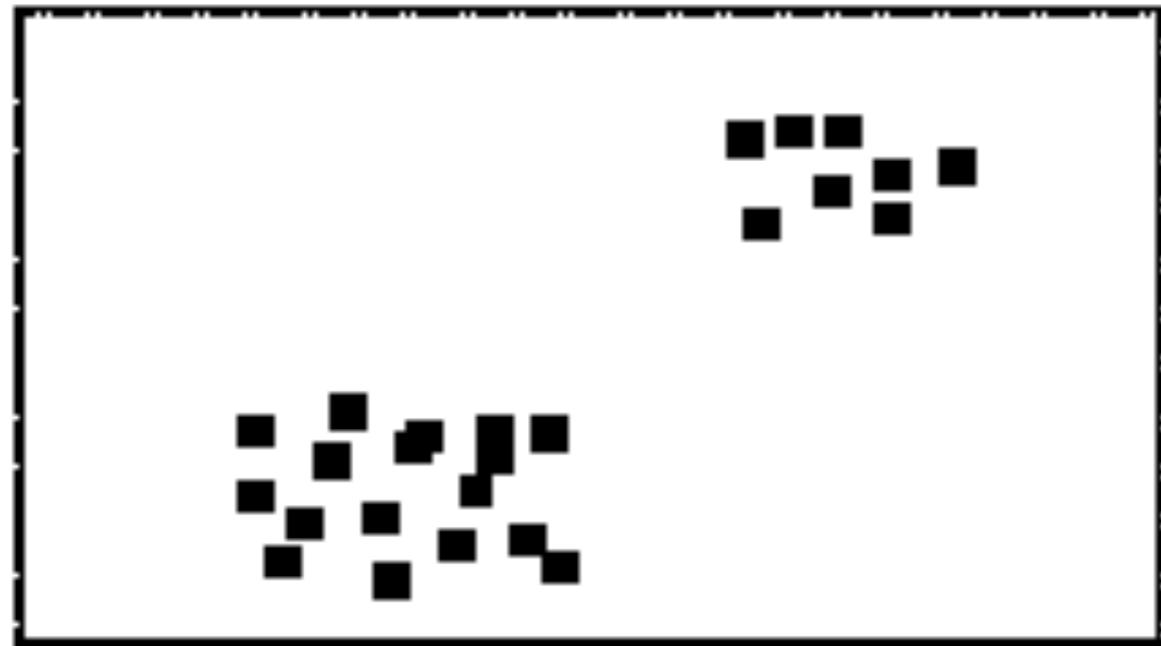
# Data regions and empty regions

- Given a data space, separate
  - data regions (clusters) and
  - empty regions (holes, with few or no data points).
- Use a supervised learning technique, i.e., decision tree induction, to separate the two types of regions.
- Due to the use of a supervised learning method for an unsupervised learning task,
  - an interesting connection is made between the two types of learning paradigms.

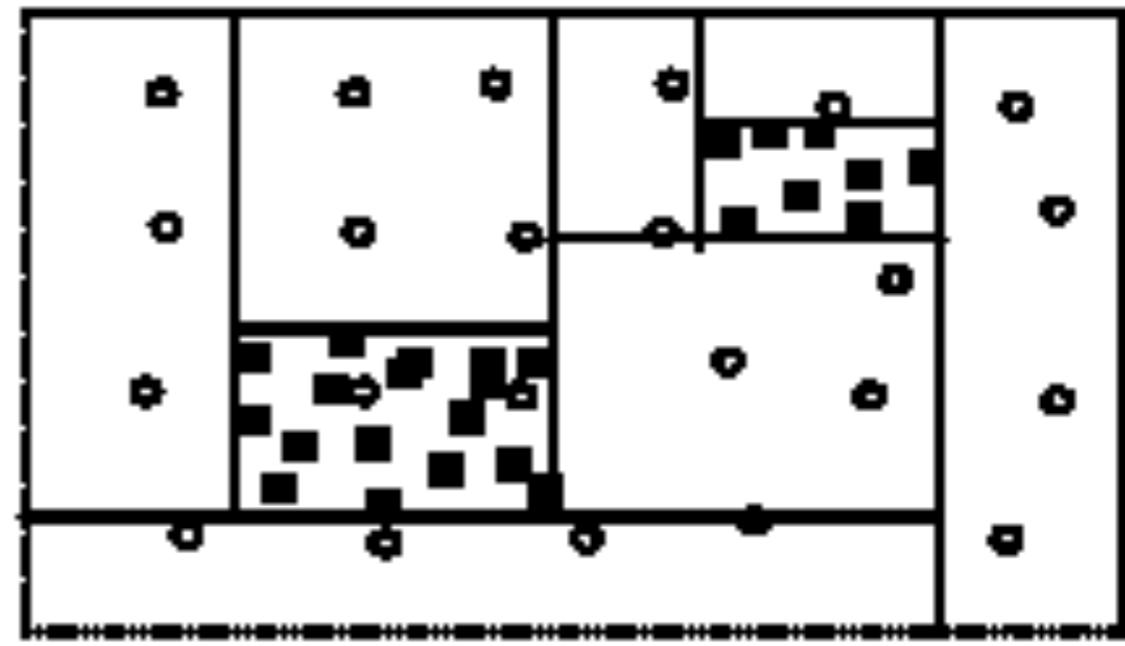
# Supervised learning for unsupervised learning

- Decision tree algorithm is not directly applicable.
  - it needs at least two classes of data.
  - A clustering data set has no class label for each data point.
- The problem can be dealt with by a simple idea.
  - Regard each point in the data set to have a class label  $Y$ .
  - Assume that the data space is uniformly distributed with another type of points, called **non-existing points**. We give them the class,  $N$ .
- With the  $N$  points added, the problem of partitioning the data space into data and empty regions becomes a supervised classification problem.

# An example



(A): The original data space



(B). Partitioning with added  
 $N$  points

- A decision tree method is used for partitioning in (B).

# Can it done without adding $N$ points?

Yes.

Physically adding  $N$  points increases the size of the data and thus the running time.

More importantly: it is unlikely that we can have points truly uniformly distributed in a high dimensional space as we would need an exponential number of points.

Fortunately, no need to physically add any  $N$  points.

We can compute them when needed

# Characteristics of the approach

It provides representations of the resulting data and empty regions in terms of **hyper-rectangles**, or **rules**.

It detects outliers automatically. Outliers are data points in an empty region.

It may not use all attributes in the data just as in a normal decision tree for supervised learning.

It can automatically determine what attributes are useful. Subspace clustering ...

**Drawback:** data regions of irregular shapes are hard to handle since decision tree learning only generates hyper-rectangles (formed by axis-parallel hyper-planes), which are rules.

# Building the Tree

The main computation in decision tree building is to evaluate **entropy** (for information gain):

$$\text{entropy}(D) = - \sum_{j=1}^{|C|} \Pr(c_j) \log_2 \Pr(c_j)$$

Can it be evaluated without adding  $N$  points? Yes.

$\Pr(c_j)$  is the probability of class  $c_j$  in data set  $D$ , and  $|C|$  is the number of classes,  $Y$  and  $N$  (2 classes).

To compute  $\Pr(c_j)$ , we only need the number of  $Y$  (data) points and the number of  $N$  (non-existing) points.

We already have  $Y$  (or data) points, and we can compute the number of  $N$  points on the fly. Simple: as we assume that the  $N$  points are uniformly distributed in the space.

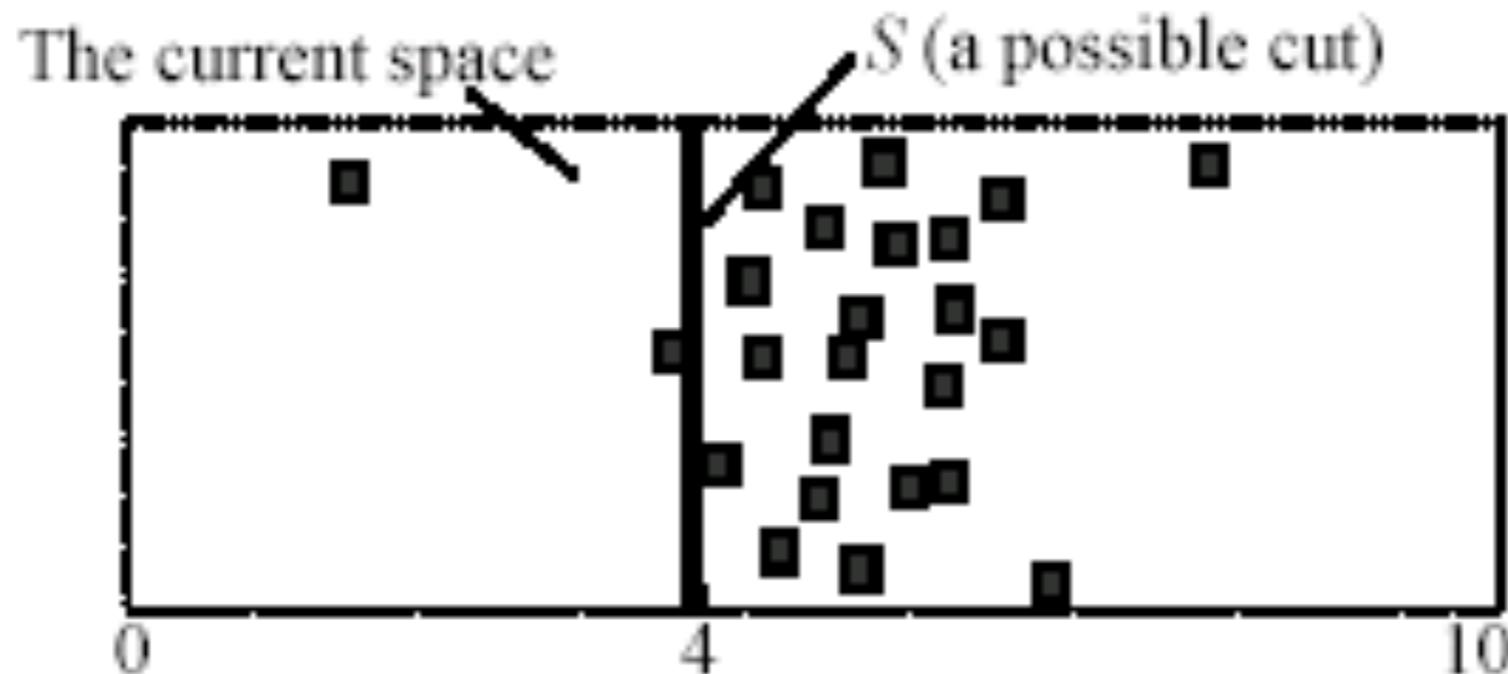
# An example

The space has 25 data ( $Y$ ) points and 25  $N$  points. Assume the system is evaluating a possible cut  $S$ .

#  $N$  points on the left of  $S$  is  $25 * 4/10 = 10$ . The number of  $Y$  points is 3.

Likewise, #  $N$  points on the right of  $S$  is 15 (= 25 - 10). The number of  $Y$  points is 22.

With these numbers, entropy can be computed.



# How many $N$ points to add?

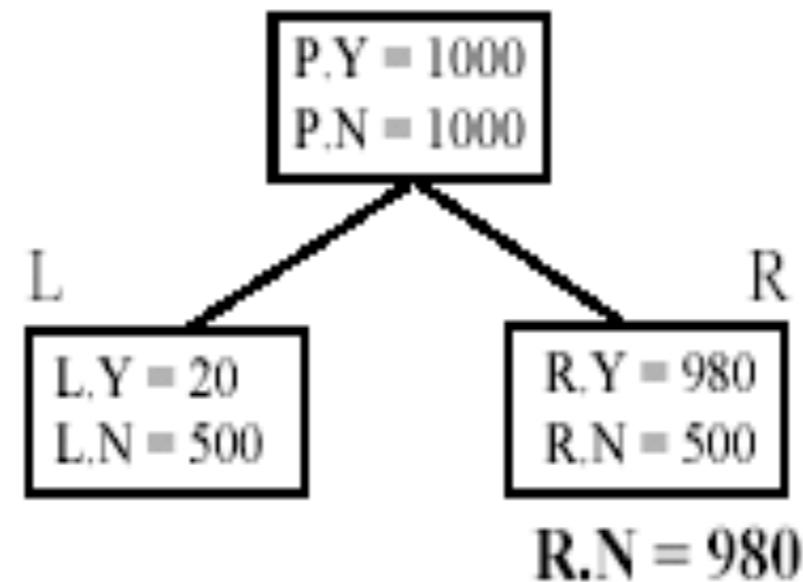
We add a different number of  $N$  points at each different node.

The number of  $N$  points for the current node  $E$  is determined by the following rule (note that at the root node, the number of inherited  $N$  points is 0):

- 1 **If** the number of  $N$  points inherited from the parent node of  $E$  is less than the number of  $Y$  points in  $E$  **then**
- 2     the number of  $N$  points for  $E$  is increased to the number of  $Y$  points in  $E$
- 3 **else** the number of inherited  $N$  points is used for  $E$

# An example

**Example 17:** Fig. 20 gives an example. The (parent) node  $P$  has two children nodes  $L$  and  $R$ . Assume  $P$  has 1000  $Y$  points and thus 1000  $N$  points, stored in  $P.Y$  and  $P.N$  respectively. Assume after splitting,  $L$  has 20  $Y$  points and 500  $N$  points, and  $R$  has 980  $Y$  points and 500  $N$  points. According to the above rule, for subsequent partitioning, we increase the number of  $N$  points at  $R$  to 980. The number of  $N$  points at  $L$  is unchanged.



# How many $N$ points to add? (cont...)

Basically, for a  $Y$  node (which has more data points), we increase  $N$  points so that

$$\#Y = \#N$$

The number of  $N$  points is not reduced if the current node is an  $N$  node (an  $N$  node has more  $N$  points than  $Y$  points).

A reduction may cause outlier  $Y$  points to form  $Y$  nodes (a  $Y$  node has an equal number of  $Y$  points as  $N$  points or more).

Then data regions and empty regions may not be separated well.

# Building the decision tree

Using the above ideas, a decision tree can be built to separate data regions and empty regions.

The actual method is more sophisticated as a few other tricky issues need to be handled in

tree building and

tree pruning.

# Summary

- Clustering is has along history and still active
  - There are a huge number of clustering algorithms
  - More are still coming every year.
- We only introduced several main algorithms. There are many others, e.g.,
  - density based algorithm, sub-space clustering, scale-up methods, neural networks based methods, fuzzy clustering, co-clustering, etc.
- Clustering is hard to evaluate, but very useful in practice. This partially explains why there are still a large number of clustering algorithms being devised every year.
- Clustering is highly application dependent and to some extent subjective.

# Recommendation System

- Concept
- Building Recommendation System

# Personalization is transforming our experience of the world



100 Hours a  
Minute

*What do I care  
about?*

## Information overload



Browsing is “history”

- Need new ways  
to discover content

Personalization: Connects *users & items*

viewers

videos

# Movie recommendations



Connect users with movies they may want to watch

# Product recommendations

The screenshot shows a portion of an Amazon website. At the top left is the Amazon logo. A red arrow points from the text "Because you purchased..." on the left to the heading "Today's Recommendations For You" on the right, which is circled in red. The "Today's Recommendations For You" section contains several book recommendations with "LOOK INSIDE" buttons. The books shown are "Even Faster Web Sites", "Simply JavaScript", and "The Art of Java". Below this section are navigation links for categories like "Any Category", "Algorithms", "Boxed Sets", "Business & Culture", "Java", "Graphic Design", "Microsoft", "Networking", "Networking, Protocols & APIs", "New", and "SQL".

Recommendations combine global & session interests

# Music recommendations



Recommendations form coherent & diverse sequence

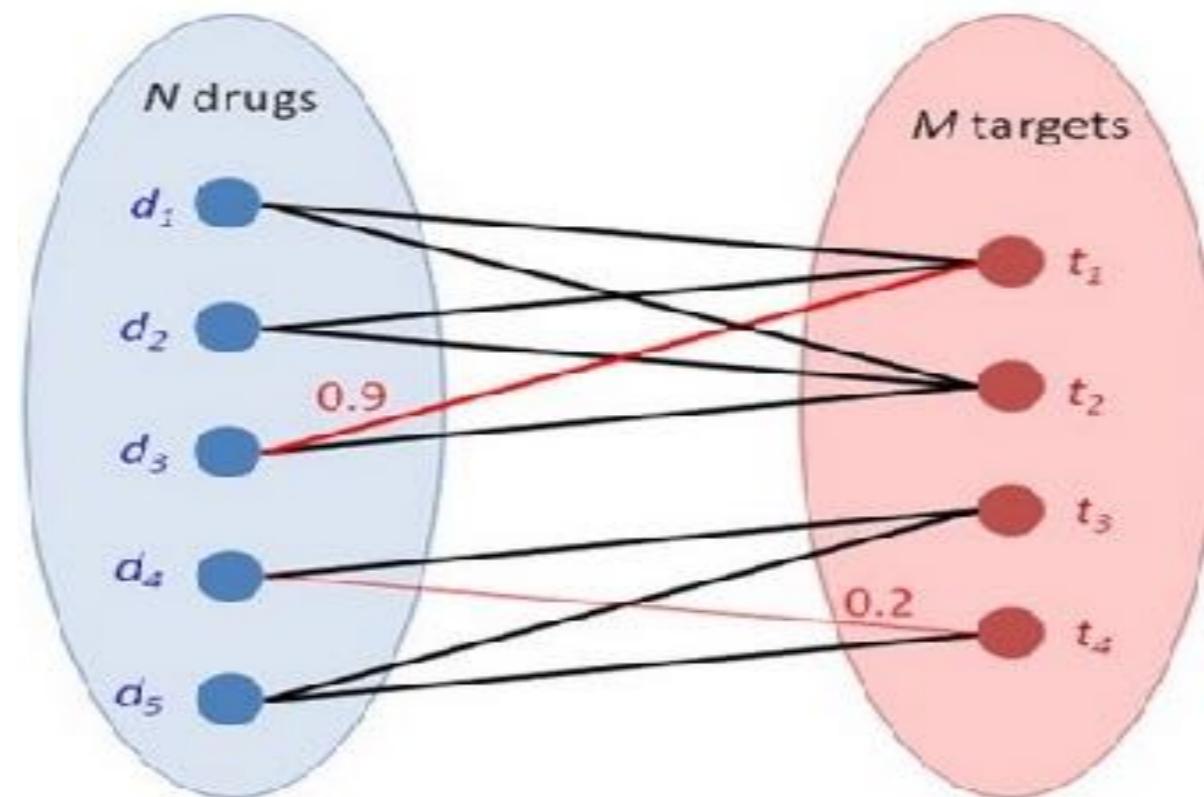
# Friend recommendations



Users and “items” are of the same “type”

# Drug-target interactions

Cobanoglu et al. '13



What drug should we “repurpose” for some disease?

# Solution 0: Popularity

# Simplest approach: Popularity

- What are people viewing now?
  - Rank by global popularity
- Limitation:
  - No personalization

MOST POPULAR

|          |         |          |
|----------|---------|----------|
| E-MAILED | BLOGGED | SEARCHED |
|----------|---------|----------|

1. Really?: The Claim: Lack of Sleep Increases the Risk of Catching a Cold.
2. Magazine Preview: Coming Out in Middle School
3. Yes, We Speak Cupcake
4. Grossamer Silk, From Spiders Spin
5. Tie to Pets Has Germ Jumping to and Fro
6. Maureen Dowd: Where the Wild Thing Is
7. Maureen Dowd: Blue Is the New Black
8. The Holy Grail of the Unconscious
9. For Opening Night at the Metropolitan, a New Sound: Booring
10. Economic Scene: Medical Malpractice System Breeds More Waste

[Go to Complete List »](#)

**CREATE YOUR HEADLINES**  
Create a personalized list of headlines based on your interests. [Get Started »](#)





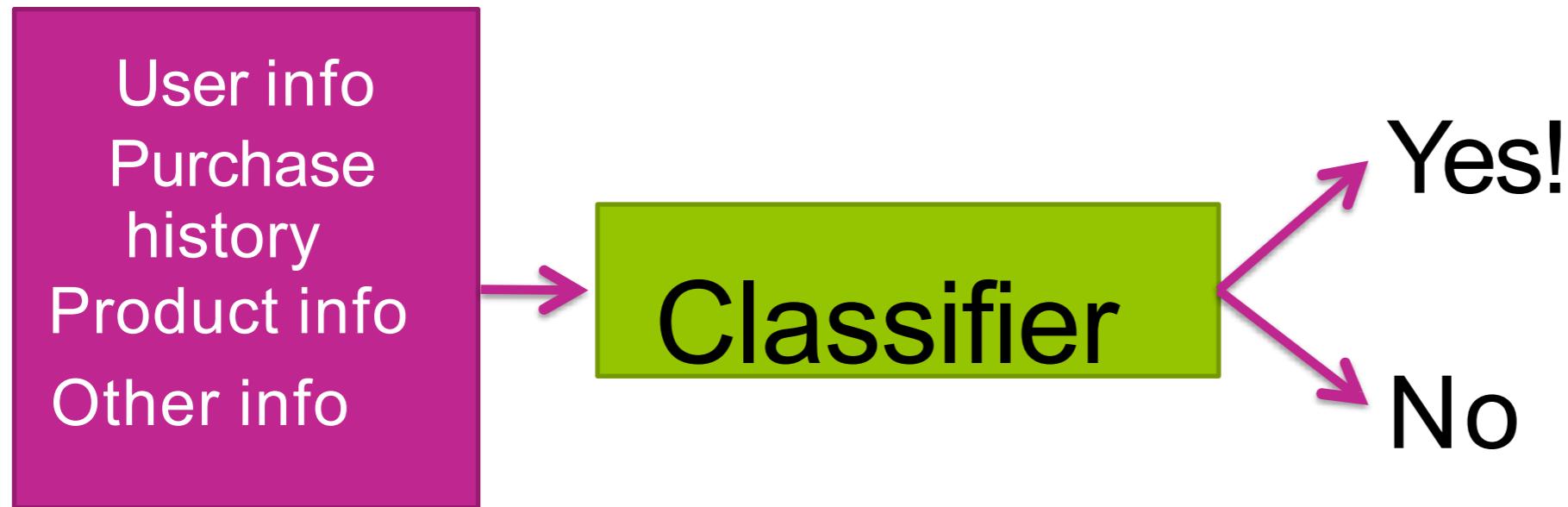
# Solution 1: Classification model

# What's the probability I'll buy this product?



- Pros:
  - **Personalized:** Considers user info & purchase history
  - **Features can capture context:** Time of the day, what I just saw,...
  - **Even handles limited user history:** Age of user, ...

# Limitations of classification approach



- Features may not be available
- Often doesn't perform as well as **collaborative filtering** methods (next)



**Solution 2: People who bought this  
also bought...**

# Co-occurrence matrix

- People who bought *diapers* also bought *baby wipes*
- Matrix C:  
store # users who bought both items  $i$  &  $j$ 
  - ( $\# \text{ items} \times \# \text{ items}$ ) matrix
  - **Symmetric:** # purchasing  $i$  &  $j$  same as # for  $j$  &  $i$   
$$(C_{ij} = C_{ji})$$

# Making recommendations using co-occurrences

- User  purchased *diapers*
  1. Look at *diapers* row of matrix
  2. Recommend other items with largest counts
    - *baby wipes, milk, baby food, ...*

# Co-occurrence matrix must be normalized

- What if there are very popular items?
  - Popular baby item:  
*Pampers Swaddlers diapers*
  - For any baby item (e.g.,  $i = \text{Sophie giraffe}$ ) large count  $C_{ij}$  for  $j = \text{Pampers Swaddlers}$
- Result:
  - Drowns out other effects
  - Recommend based on popularity



# Normalize co-occurrences: Similarity matrix

- Jaccard similarity: normalizes by popularity
  - Who purchased  $i$  and  $j$  divided by who purchased  $i$  or  $j$
- Many other similarity metrics possible, e.g., cosine similarity

# Limitations

- Only current page matters, **no history**
  - Recommend similar items to the one you bought
- What if you purchased many items?
  - Want recommendations based on purchase history

# (Weighted) Average of purchased items

- User  bought items  $\{diapers, milk\}$ 
  - Compute user-specific score for each item  $j$  in inventory by combining similarities:

$$\begin{aligned} \text{Score}(\text{User}, \text{baby wipes}) &= \\ \frac{1}{2} (S_{\text{baby wipes}, \text{diapers}} + S_{\text{baby wipes}, \text{milk}}) \end{aligned}$$

- Could also weight recent purchases more
- Sort  $\text{Score}(\text{User}, j)$  and find item  $j$  with highest similarity

# Limitations

- Does not utilize:
  - **context** (e.g., time of day)
  - **user features** (e.g., age)
  - **product features** (e.g., baby vs. electronics)
- Cold start problem
  - What if a new user or product arrives?



## Solution 3: Discovering hidden structure by matrix factorization

# Movie recommendation

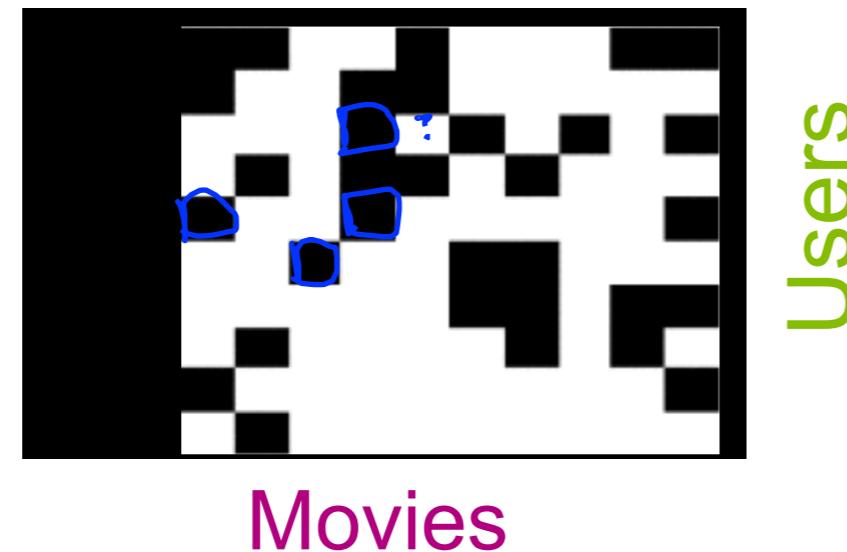
- Users watch movies and rate them

| User   | Movie   | Rating    |
|--------|---------|-----------|
| User 1 | Movie 1 | ★ ★ ★ ★ ★ |
| User 1 | Movie 2 | ★ ★ ★ ★ ★ |
| User 1 | Movie 3 | ★ ★ ★ ★ ★ |
| User 2 | Movie 1 | ★ ★ ★ ★ ★ |
| User 2 | Movie 2 | ★ ★ ★ ★ ★ |
| User 2 | Movie 3 | ★ ★ ★ ★ ★ |
| User 3 | Movie 1 | ★ ★ ★ ★ ★ |
| User 3 | Movie 2 | ★ ★ ★ ★ ★ |
| User 3 | Movie 3 | ★ ★ ★ ★ ★ |
| User 4 | Movie 1 | ★ ★ ★ ★ ★ |
| User 4 | Movie 2 | ★ ★ ★ ★ ★ |
| User 4 | Movie 3 | ★ ★ ★ ★ ★ |
| User 5 | Movie 1 | ★ ★ ★ ★ ★ |
| User 5 | Movie 2 | ★ ★ ★ ★ ★ |
| User 5 | Movie 3 | ★ ★ ★ ★ ★ |
| User 6 | Movie 1 | ★ ★ ★ ★ ★ |
| User 6 | Movie 2 | ★ ★ ★ ★ ★ |
| User 6 | Movie 3 | ★ ★ ★ ★ ★ |
| User 7 | Movie 1 | ★ ★ ★ ★ ★ |
| User 7 | Movie 2 | ★ ★ ★ ★ ★ |
| User 7 | Movie 3 | ★ ★ ★ ★ ★ |
| User 8 | Movie 1 | ★ ★ ★ ★ ★ |
| User 8 | Movie 2 | ★ ★ ★ ★ ★ |
| User 8 | Movie 3 | ★ ★ ★ ★ ★ |
| User 9 | Movie 1 | ★ ★ ★ ★ ★ |
| User 9 | Movie 2 | ★ ★ ★ ★ ★ |
| User 9 | Movie 3 | ★ ★ ★ ★ ★ |

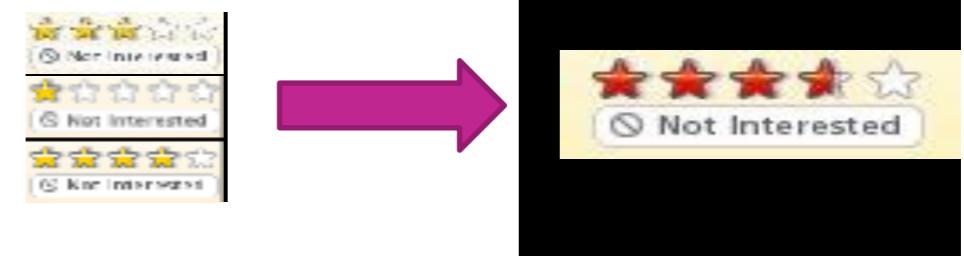
Each user only watches a few of the available movies

# Matrix completion problem

Rating =



- Data: Users score some movies  
 $\text{Rating}(u, v)$  known for black cells  
 $\text{Rating}(u, v)$  unknown for white cells
- Goal: Filling missing data?

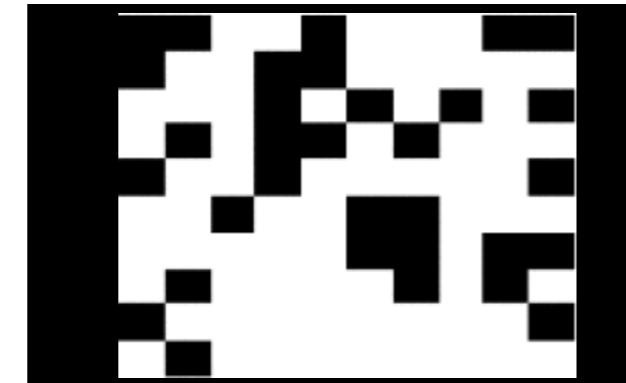


# Suppose we had d topics for each user and movie

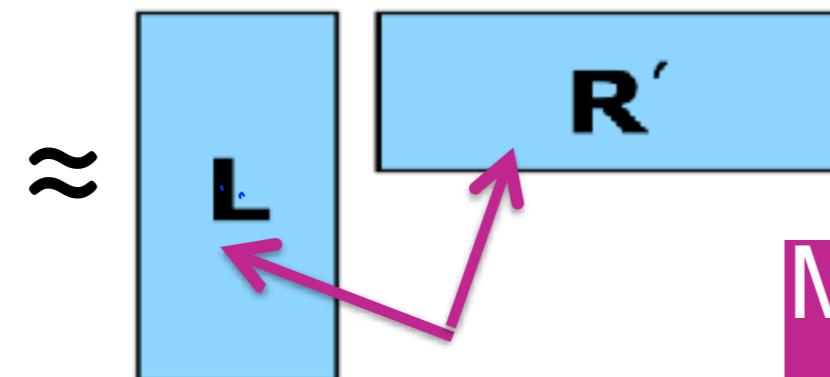
- Describe movie  $v$   with topics  $R_v$ 
  - How much is it **action, romance, drama,...**
- Describe user  $u$   with topics  $L_u$ 
  - How much she likes **action, romance, drama,...**
- $\text{Rating}(u, v)$  is the product of the two vectors
- Recommendations: sort movies user hasn't watched by  $\text{Rating}(u, v)$

# Matrix factorization model: Discovering topics from data

Rating =



Parameters of model



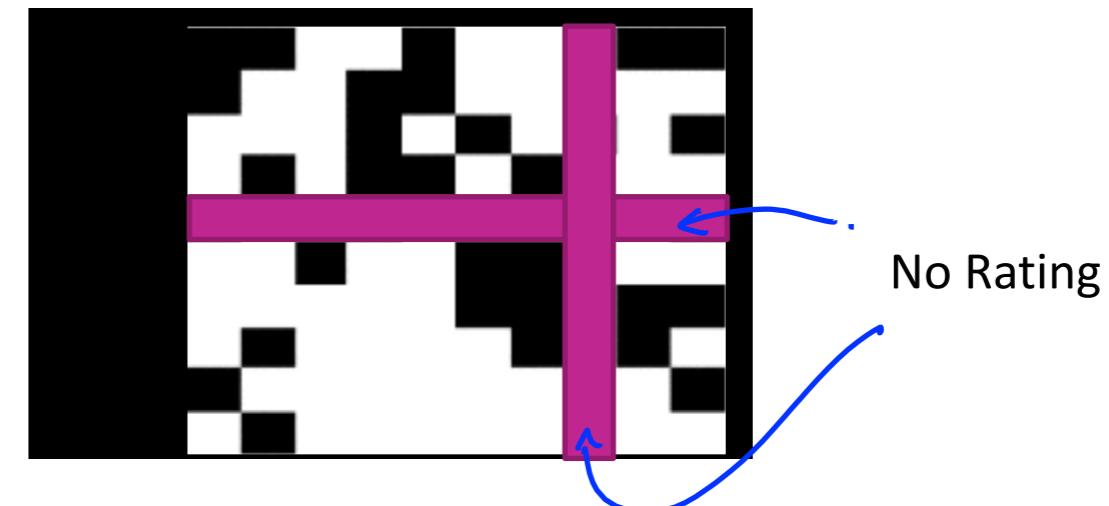
Many efficient algorithms for factorization

- Only use observed values to estimate “topic” vectors  $\hat{L}_u$  and  $\hat{R}_v$
- Use estimated  $\hat{L}_u$  and  $\hat{R}_v$  for recommendations

# Limitations of matrix factorization

- Cold-start problem
  - This model still cannot handle a new user or movie

Rating =





# Bringing it all together: Featurized matrix factorization

# Combining features and discovered topics

- Features capture **context**
  - *Time of day, what I just saw, user info, past purchases,...*
- Discovered topics from matrix factorization capture **groups of users** who behave similarly
  - *Women from Seattle who teach and have a baby*
- **Combine** to mitigate cold-start problem
  - Ratings for a new user from **features** only
  - As more information about user is discovered, matrix factorization **topics** become more relevant

# Blending models

- Squeezing last bit of accuracy by blending models
- Netflix Prize 2006-2009
  - 100M ratings
  - 17,770 movies
  - 480,189 users
  - Predict 3 million ratings to highest accuracy
  - Winning team blended over 100 models



The screenshot shows the Netflix Prize Leaderboard. The top banner displays "Netflix Prize" and the current leader's score of "10.05%". Below the banner is a table with columns: Rank, Team Name, Best Score, % Improvement, and Last Submit Time. The table lists the top 8 teams, with the first team being the winning team. A yellow arrow points to the "% Improvement" column for the winning team.

| Rank                         | Team Name                 | Best Score | % Improvement | Last Submit Time    |
|------------------------------|---------------------------|------------|---------------|---------------------|
| 1                            | Bellkor's Pragmatic Chaos | 0.8988     | 10.05         | 2009-06-26 18:42:37 |
| Grand Prize - RMSE <= 0.8962 |                           |            |               |                     |
| 2                            | PragmaticTheory           | 0.8968     | 9.80          | 2009-06-26 22:10:51 |
| 3                            | Bellkor in BigChaos       | 0.8960     | 9.71          | 2009-06-13 09:14:09 |
| 4                            | Grand Prize Team          | 0.8958     | 9.58          | 2009-06-12 09:20:24 |
| 5                            | Dase                      | 0.8604     | 9.55          | 2009-04-22 05:57:03 |
| 6                            | RegChaos                  | 0.8613     | 9.47          | 2009-06-23 23:08:52 |



# A performance metric for recommender systems

# The world of all baby products



# User likes subset of items



# Why not use classification accuracy?

- Classification accuracy = fraction of items correctly classified (*liked* vs. *not liked*)
- Here, not interested in what a person *does not like*
- Rather, how quickly can we discover the relatively few *liked* items?
  - (Partially) an imbalanced class problem

# How many liked items were recommended?



# How many recommended items were liked?



# Maximize recall: Recommend everything



# Resulting precision?



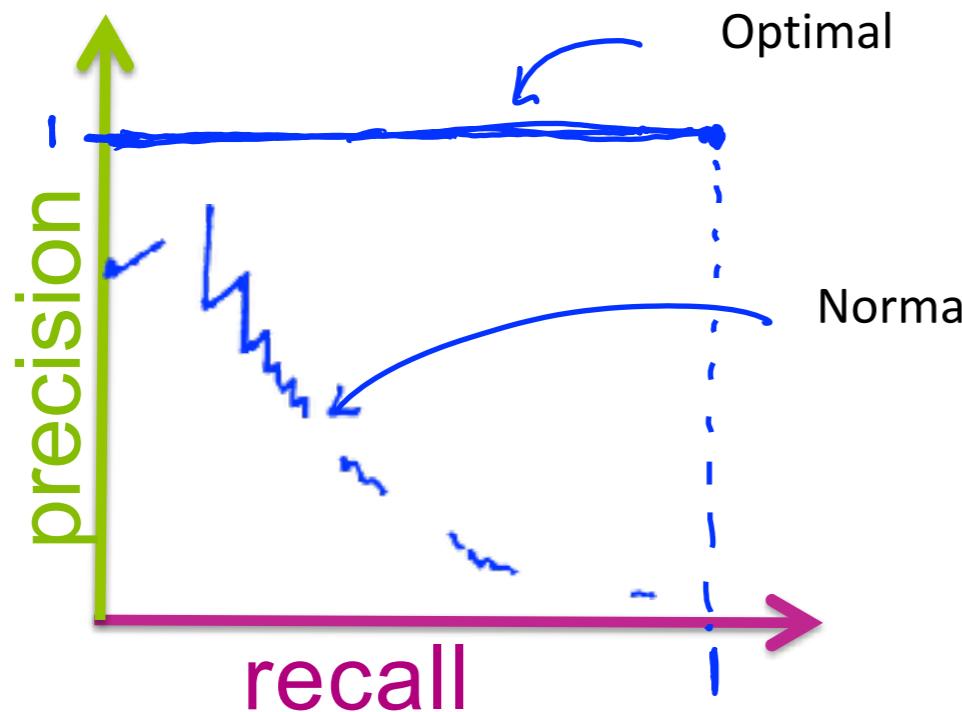
Precision  
# liked & shown  
\_\_\_\_\_  
# shown

# Optimal recommender



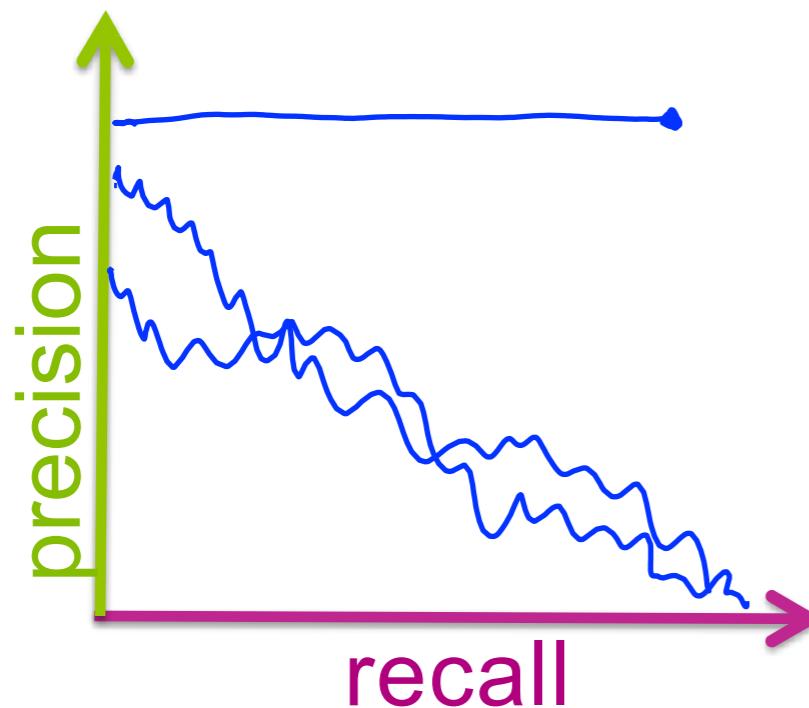
# Precision-recall curve

- Input: A specific recommender system
- Output: Algorithm-specific precision-recall curve
- To draw curve, vary threshold on # items recommended
  - For each setting, calculate the precision and recall



# Which Algorithm is Best?

- For a given **precision**, want **recall** as large as possible (or vice versa)
- One metric: largest **area under the curve (AUC)**
- Another: set desired recall and maximize precision (**precision at k**)



# Summary of Recommender System

- Describe the goal of a recommender system
- Provide examples of applications where recommender systems are useful
- Implement a co-occurrence based recommender system
- Describe the input (observations, number of “topics”) and output (“topic” vectors, predicted values) of a matrix factorization model
- Exploit estimated “topic” vectors (algorithms to come...) to make recommendations
- Describe the cold-start problem and ways to handle it (e.g., incorporating features)
- Analyze performance of various recommender systems in terms of precision and recall
- Use AUC or precision-at-k to select amongst candidate algorithms

# Neural Network

- Neural networks learning methods provide a robust approach to approximating real-valued, discrete valued and vector valued target functions.
- Often used in problems such as handwriting, voice, or image recognition
- Feed-Forward Neural Networks, Convolutional Neural Networks, Deep Learning

# Biological Motivation

- Inspired by biological learning systems that are built from very complex webs of interconnected neurons

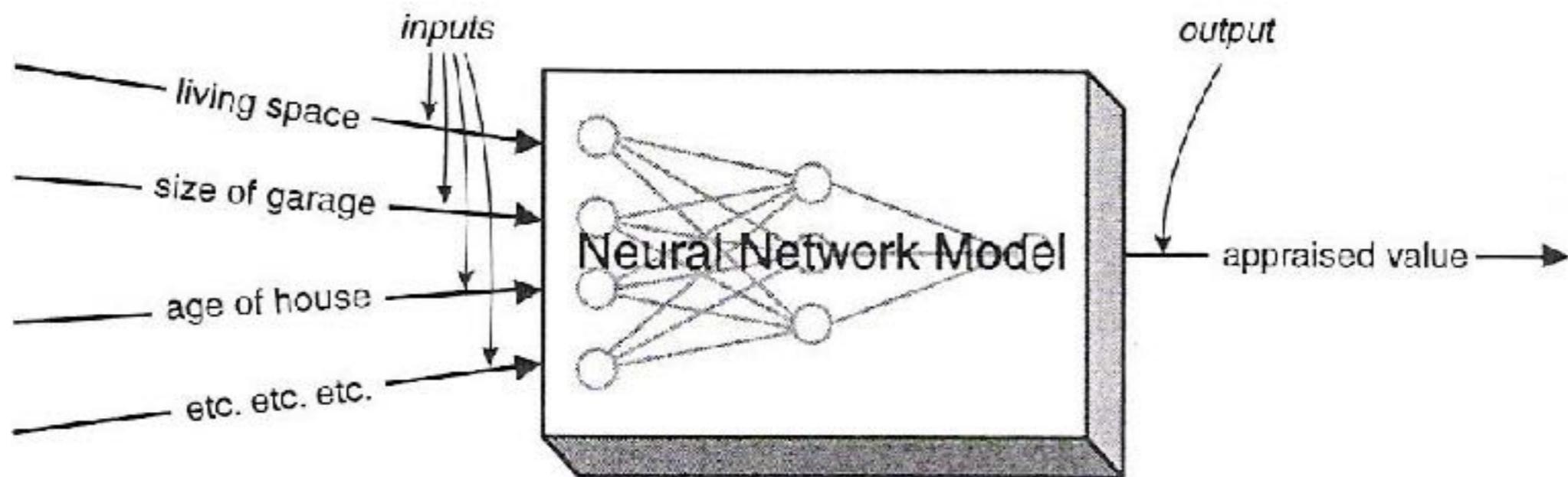
# Facts About Neuro-biology

- Human brain is estimated to contain a densely interconnected network of approximately  $10^{11}$  neurons.
- Each connected, on average, to  $10^4$  other neurons.
- Neuron activity is typically excited through connections to other neurons.
- The fastest neuron switching times are in the order of  $10^{-3}$  seconds.
- It requires  $10^{-1}$  seconds to visibly recognize your mother.

# Analogy of Biological Learning Systems

- Artificial neural networks are built out of a interconnected set of simple units, where each unit takes a number of real-valued inputs (can be the outputs of other units) and produces a single real-valued output (which may become the input of other units).

# Example

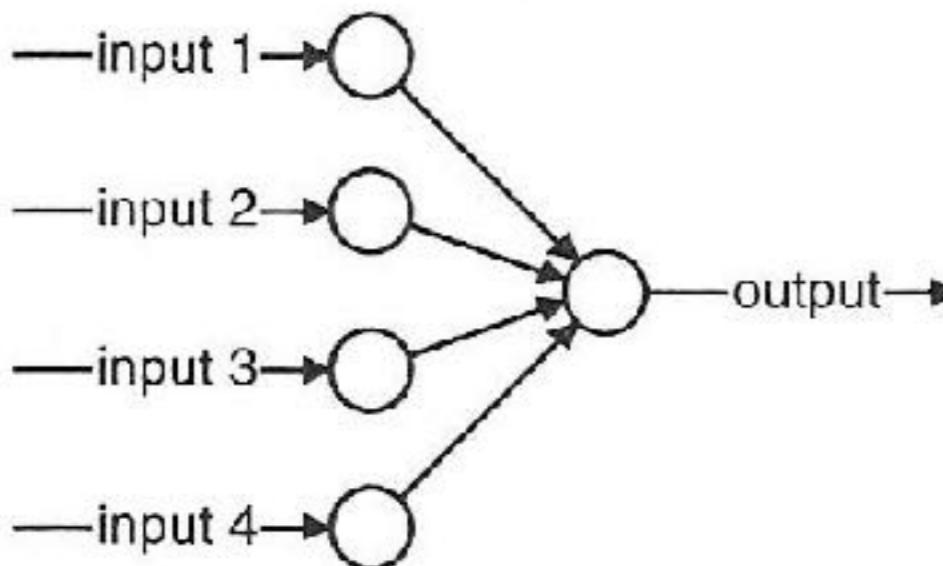


A neural network is like a black box that know how to process input to create an output. The calculation is quite complex and difficult to understand.

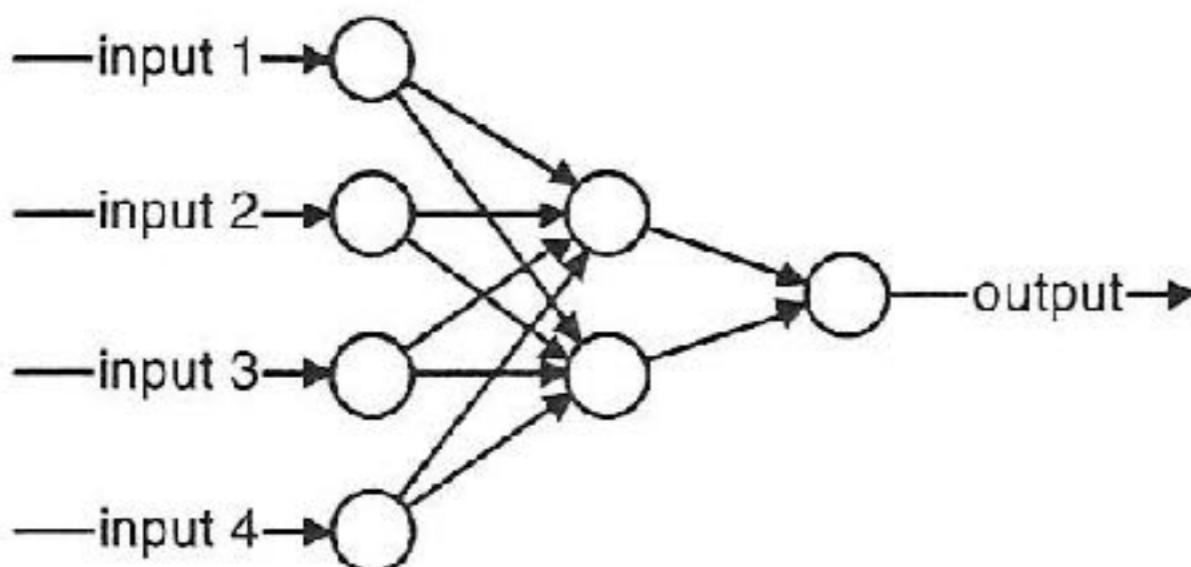
# Neural Networks for Directed Data Mining

1. Identify the input and output features
2. Transform the input and output so that they are in a small range (**-1 to 1**)
3. Set up a network with an appropriate topology
4. Train the network on a representative set of training examples
5. Use the validation set to choose the set of weights that minimizes the error
6. Evaluate the network using the test set to see how well it performs
7. Apply the model generated by the network to predict outcomes for unknown input

# What is a Neural Network? (1)

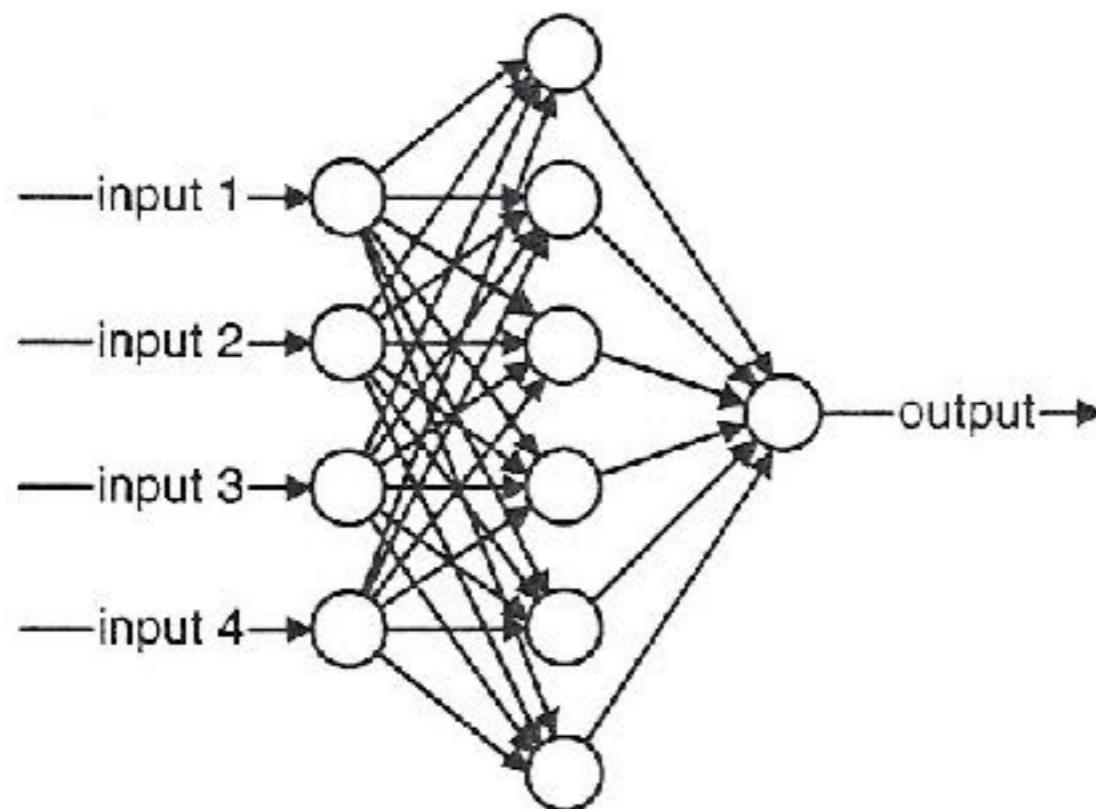


This simple neural network takes four inputs and produces an output. This result of training this network is equivalent to the statistical technique called logistic regression.

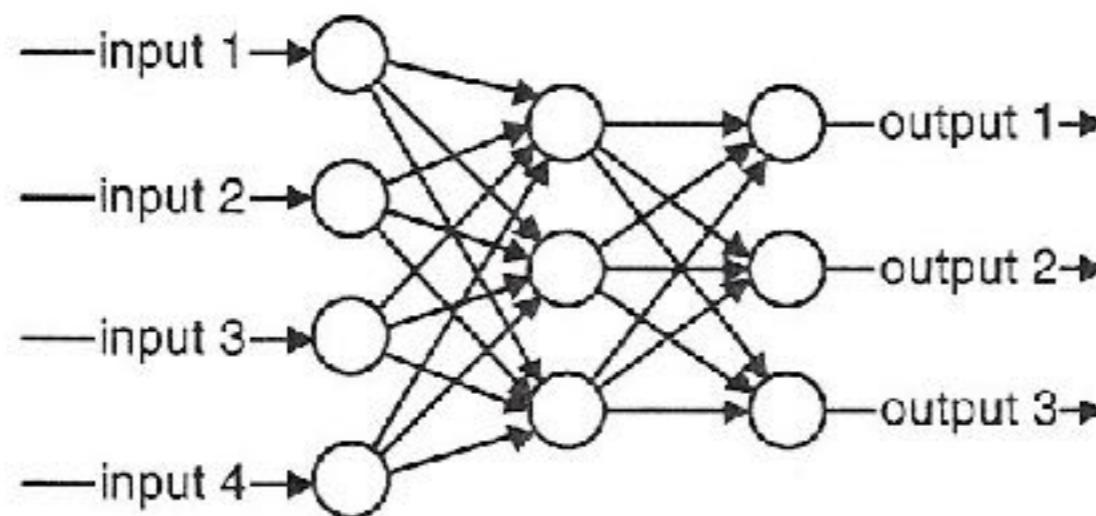


This network has a middle layer called the *hidden layer*, which makes the network more powerful by enabling it to recognize more patterns.

# What is a Neural Network? (2)



Increasing the size of the hidden layer makes the network more powerful but introduces the risk of overfitting. Usually, only one hidden layer is needed.



A neural network can produce multiple output values.

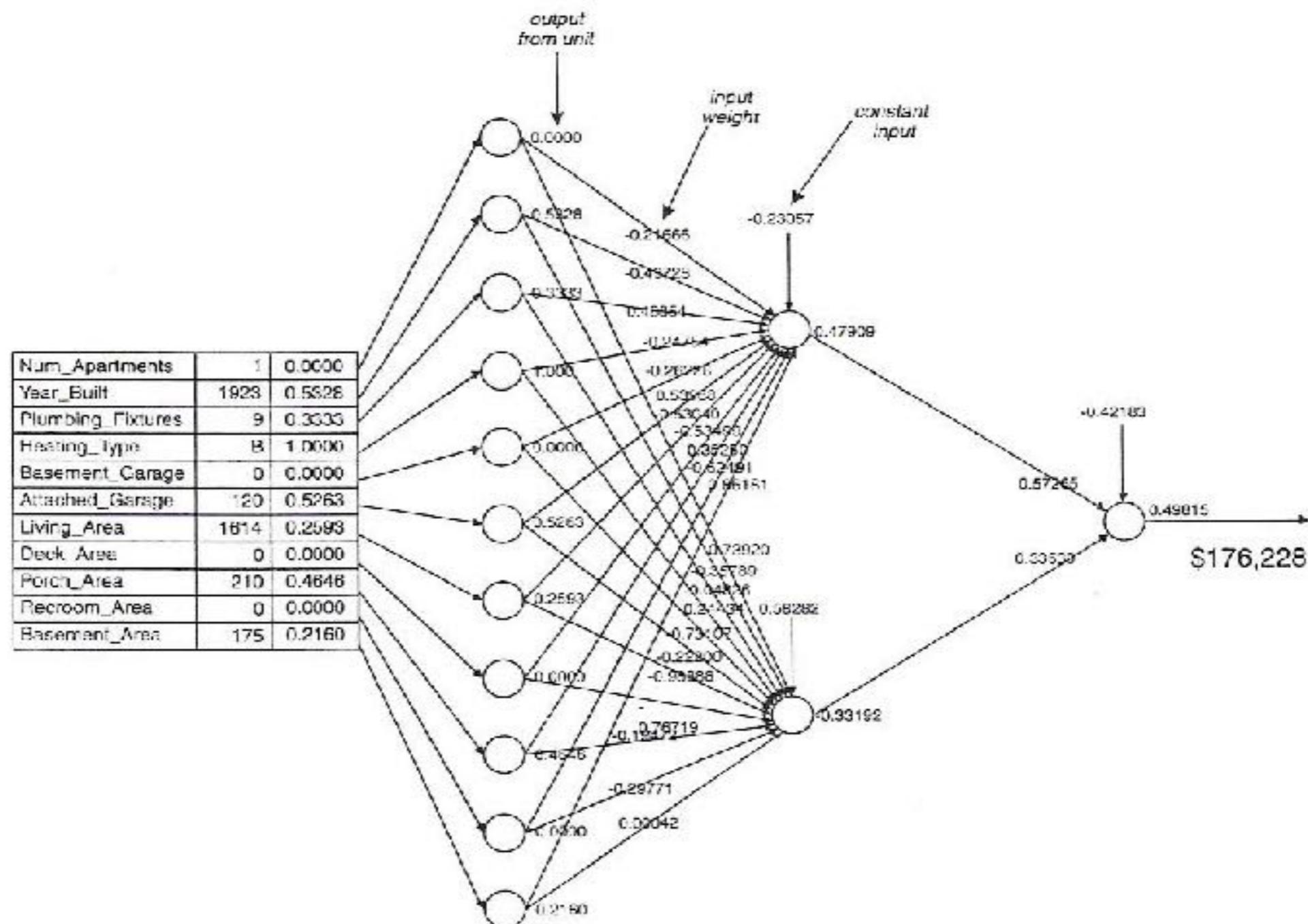
# Three Main Components of a Neural Network

- Activation function
- Network topology
- How the network is trained?
  - Feed forward
  - Back propagation

# Activation Function (1)

- Combination function
  - The function that merges all the input into a single value.
  - The most common combination function is the weighted sum.
- Transfer function
  - The function that transfers the value of the combination function to the output of the unit.

# Feed-Forward Neural Networks (1)



# Feed-Forward Neural Networks: Input Layer

- Input layer
- Each unit in the input layer is connected to exactly one source field.

# Feed-Forward Neural Networks : Hidden layer (1)

- Hidden layer
- Each unit in the hidden layer is typically connected to all the units in the input layer.
- The units in the hidden layer calculate their output by multiplying the value of each input by its corresponding weight, adding them up, and applying the transfer function.

# Feed-Forward Neural Networks: Hidden layer (2)

- Hidden layer
- A neural network can have any number of hidden layers, but in general **one** hidden layer is sufficient.
- The wider the layer (the more units it contains), the greater the capacity of the network to recognize patterns.

# Feed-Forward Neural Networks : Output Layer (1)

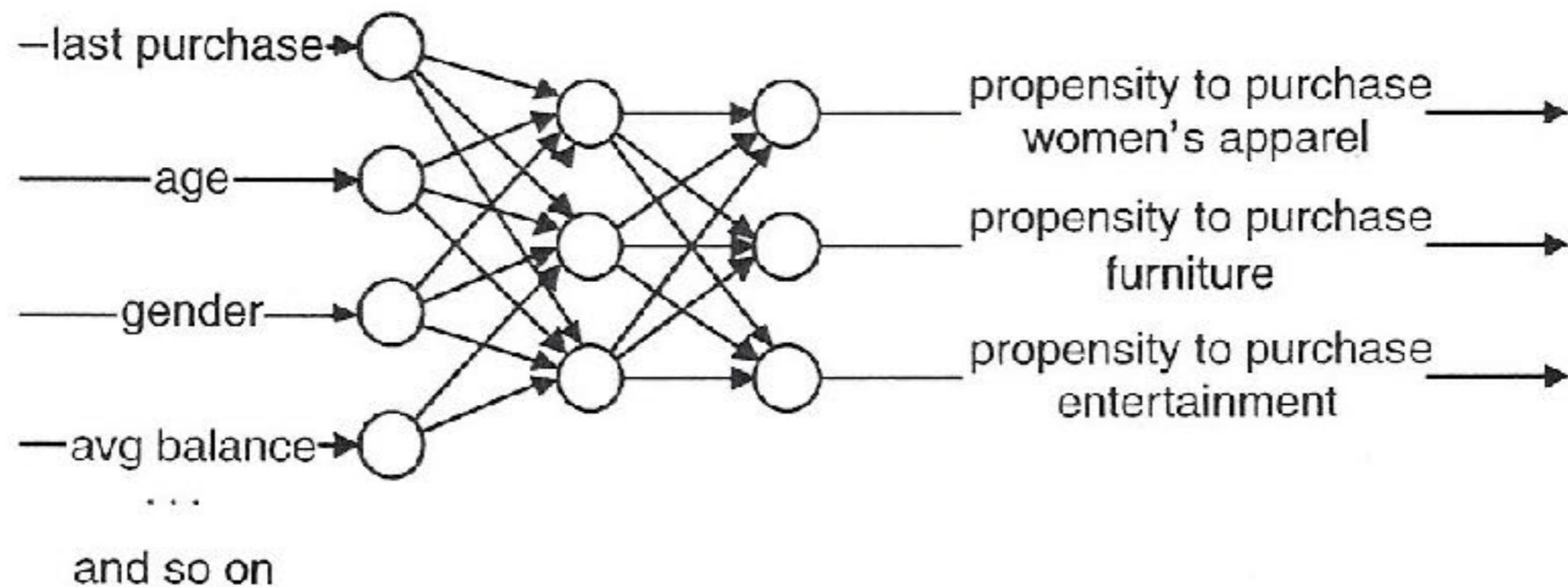
## Output layer

- It is fully connected to all the units in the hidden layer.
- Most of the time, the neural network is being used to calculate a single value, so there is only one unit in the output layer and the value.
- We must map this value back to understand the output.

# Feed-Forward Neural Networks: Output Layer (2)

- Output layer
- It is possible for the output layer to have more than one unit.
- Eg. A department store chain wants to predict the likelihood that customers will be purchasing products from various departments, such as women's apparel, furniture, and entertainments.

# Feed-Forward Neural Networks: Output Layer (3)



- After feeding the inputs for a customer into the network, the network calculates three values.

# Back-Propagation Algorithm

- Assume the network is a fixed structure that corresponds to a directed graph.
- Learning corresponds to choosing a weight value for each edge in the graph.
- It attempts to minimize the squared error between the network output values and the target values for these outputs.
- It is the ANN learning technique that is most commonly used.

# Back Propagation Neural Networks: Steps (1)

1. The network gets a training example and, using the existing weights in the network, it calculates the output.
2. Back propagation calculates the error by taking the difference between the calculated result and the expected (actual result).

# Back Propagation Neural Networks: Steps (2)

3. The error is fed back through the network and the weights are adjusted to minimize the error.
4. After being shown enough training examples, the weights on the network no longer change significantly and the error no longer decreases. This is the point where training stops.

# Back-Propagation Algorithm: Input

- Training example is a pair of the form  $\langle x, t \rangle$ , where  $x$  is the vector of network input values and  $t$  is the vector of target network output values
- $n$  is the learning rate
- $n_{in}$  is the number of network inputs
- $n_{hidden}$  is the number of units in the hidden layer
- $n_{out}$  is the number of output units
- The input from unit  $i$  into unit  $j$  is denoted  $x_{ji}$
- The weight from unit  $i$  to unit  $j$  is denoted  $w_{ji}$

# Back-Propagation Algorithm: process (1)

- Create a feed-forward network with  $n_{in}$  inputs,  $n_{hidden}$  hidden units, and  $n_{out}$  output units
- Initialize all network weight to small random numbers
- Until the termination condition is met, Do

# Back-Propagation Algorithm: process (2)

Propagate the input forward through the network:

1. Input the instance  $x$  to the network and compute the output  $o_u$  of every unit  $u$  in the network

Propagate the errors backward through the network:

2. For each network output unit  $k$ , calculate its error term  $e_k$

$$e_k \leftarrow o_k (1 - o_k)(t_k - o_k)$$

# Back-Propagation Algorithm: process

3. For each hidden unit  $h$ , calculate its error term  $e_h$

$$e_h \leftarrow o_h(1 - o_h) \sum_{k \in outputs} w_{kh} e_k$$

4. Update each network weight  $w_{ji}$

$$w_{ji} \leftarrow w_{ji} + \Delta w_{ji}$$

where

$$\Delta w_{ji} = n e_j x_{ji}$$

# Termination conditions

- Stop after a fixed number of iterations
- Stop when the error on the training examples falls below some threshold
- Stop when the error on a separate validation set of examples meets some criterion

# Hidden Layer Representations

- Training examples constrain the network inputs and outputs.
- The weight tuning procedure is free to set weights that define whatever hidden unit representation is most effective in minimizing the squared error.
- The ability of multilayer networks to automatically discover useful representation **at the hidden layers** is a key feature of ANN learning.

# Heuristics for Using Feed-Forward & Back Propagation Networks

- One important decision is the **number of units in the hidden layer**. The more unit, the more patterns the network can recognize.
- We generally do not use hidden layers larger than the number of inputs.
- A good place to start for many problems is to experiment with one, two, and three nodes in the hidden layer.

## Choosing the Training Set: Coverage of values for all Features

- In general, it is good to have several examples in the training set for each value of a categorical feature and for values throughout the ranges of ordered discrete and continuous features.

# Choosing the Training Set: Number of Features

- The more features used as inputs into the networks, the larger the networks needs to be, increasing the risk of overfitting and increasing the size of the training set.
- The more features, the longer it takes the network to converge to a set of weights.
- In many cases, it is useful to calculate new variables that represent particular aspects of the business problems.

# Choosing the Training Set: Size of Training Set

- The more features there are, the more training examples that are needed to get a good coverage of patterns in the data.
- Overfitting is guaranteed to happen when there are fewer training examples than there are weights in the network.

# Choosing the Training Set: Number of Output

- The number of training examples for each possible output should be about the same.

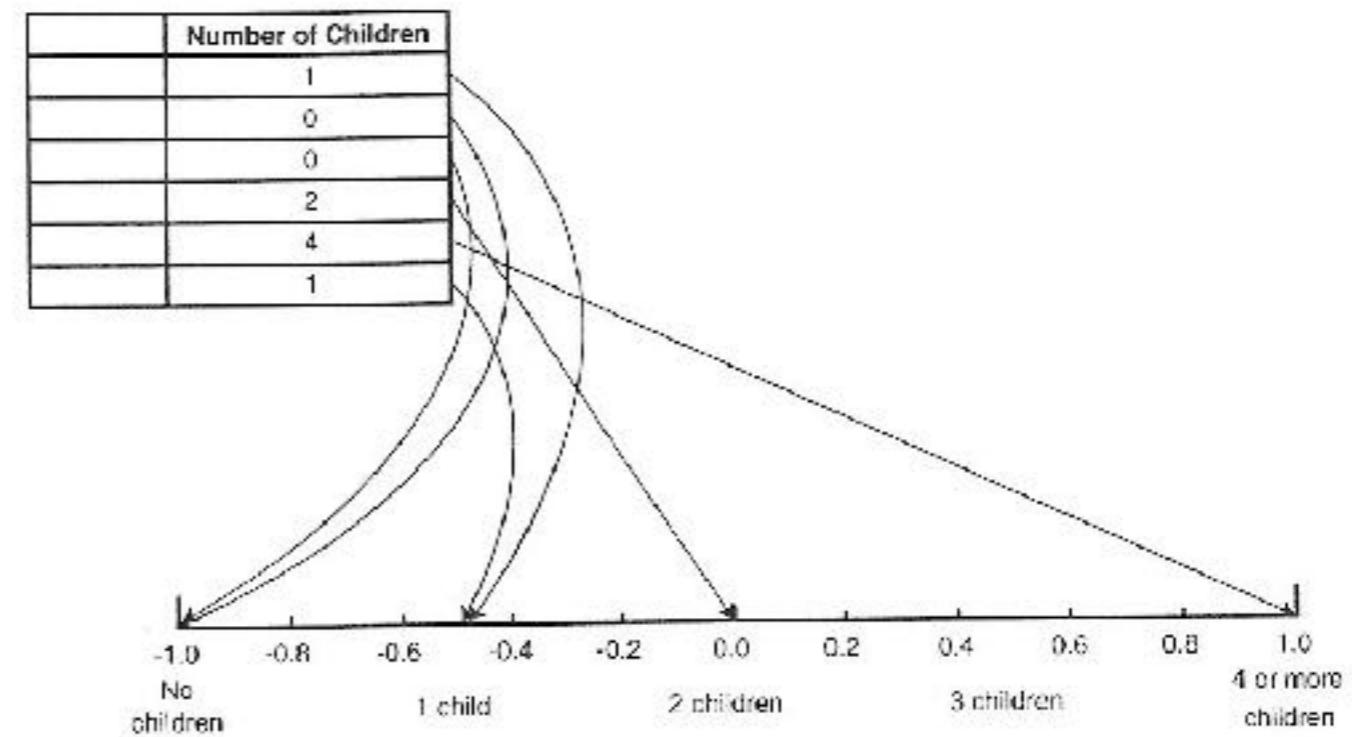
# Preparing the Data: Features with Continuous Values

- The value should be scaled to be in a reasonable range.
- Plan for a larger range
- Reject out-of-range values
- Peg values lower than the minimum to the minimum and higher than the maximum to the maximum
- Map the minimum value to  $-0.9$  and the maximum value to  $0.9$  instead of  $-1$  and  $1$

# Preparing the Data: Features with Ordered Discrete Values (1)

- First, count the number of different values and assign each a proportional fraction into some range.
- eg. If there are five distinct values, these get mapped to 0, 0.25, 0.50, 0.75, and 1.0

$$\begin{array}{rcl} 0 & \rightarrow & 0\ 0\ 0\ 0 = 0/16 = 0.0000 \\ 1 & \rightarrow & 1\ 0\ 0\ 0 = 8/16 = 0.5000 \\ 2 & \rightarrow & 1\ 1\ 0\ 0 = 12/16 = 0.7500 \\ 3 & \rightarrow & 1\ 1\ 1\ 0 = 14/16 = 0.8750 \end{array}$$



# Preparing the Data: Features with Ordered Discrete Values (2)

- It is also possible to break a range into unequal parts.
- One example is called thermometer codes

$$\textcircled{m} \ 0 \rightarrow 0\ 0\ 0 = 0 / 16 = 0.0000$$

$$\textcircled{m} \ 1 \rightarrow 1\ 0\ 0 = 8 / 16 = 0.5000$$

$$\textcircled{m} \ 2 \rightarrow 1\ 1\ 0 = 12 / 16 = 0.7500$$

$$\textcircled{m} \ 3 \rightarrow 1\ 1\ 1 = 14 / 16 = 0.8750$$

It shows that the difference on one end of the scale is less significant than differences on the other end.

# Preparing the Data: Features with Categorical Values (1)

- When working with categorical variables in neural networks, the mapping of variables into numbers **introduces an ordering** of the variables, which the neural network takes into account.
- The second way of handling categorical features is to break the categories into flags, one for each value.

# Preparing the Data: Features with Categorical Values (2)

| Gender  | N coding         |                    |                     | N-1 coding       |                    |
|---------|------------------|--------------------|---------------------|------------------|--------------------|
|         | Gender Male Flag | Gender Female Flag | Gender Unknown Flag | Gender Male Flag | Gender Female Flag |
| Male    | +1.0             | -1.0               | -1.0                | +1.0             | -1.0               |
| Female  | -1.0             | +1.0               | -1.0                | -1.0             | +1.0               |
| Unknown | -1.0             | -1.0               | +1.0                | -1.0             | -1.0               |

# Interpreting the Results (1)

- When estimating a continuous value, often the output needs to be scaled back to the correct range.
- For binary or categorical output variables, the approach is still to take the inverse of the translation used for training the network.
- eg. If “churn” is given the value of 1 and “no churn” a value of  $-1$ , the values near 1 represent churn and those near  $-1$  represent no churn.

# Interpreting the Results(2)

- When there are two outcomes, the meaning of the output depends on the training set used to train the network.
- The average value produced by the network during training is usually going to be close to the average value in the training set.

# Interpreting the Results (3)

- One way to handle is, eg.
- If the training set had 50% churn and 50% no churn, the average value the network will produce from the training examples is going to be close to 0.0.
- Values higher than 0.0 are more like churn and those less than 0.0, less like churn.
- If the original training set had 10% churn, then the cutoff would more reasonable be –
- 0.8 rather than 0.0 (0.8 is 10% of the way from  $-1$  to  $1$ ).

# Interpreting the Results (4)

- For binary values, it is also possible to create a network that produces two output, one for each value.
- In this case, each output represents the strength of evidence that that category is the correct one. The chosen category would then be the one with the higher value.

# How to Know What is Going on Inside a Neural Network

Sensitivity analysis uses the test set to determine how sensitive the output of the network is to each input :

1. Find the average value for each input.
2. Measure the output of the network when all inputs are at their average value.
3. Measure the output of the network when each input is modified, one at a time, to be at **its minimum and maximum values**.

# Summary

- Neural networks can be used for both categorical and continuous inputs.
- Neural networks learn best when input fields have been mapped to the range between -1 and +1.
- Neural networks work best when there are only a few variables.
- When training a network, there is no guarantee that the resulting set of weights is optimal.
- A neural network cannot explain what it is doing.



# Practical Machine Learning for Data Science

Fundamental Data Science for Data Scientist

# Overview

- Predictive Data Mining
  - Two Phases of Processing
    - Training Phase : Learn a model from training data
    - Predicting Phase : Deploy the model to production and use that to predict the future outcome

# Data

Iris Data Set from UCI Machine Learning Repository (<https://archive.ics.uci.edu/ml/datasets/Iris>)

Attribute Information:

1. Sepal Length in cm
2. Sepal width in cm
3. Petal length in cm
4. Petal width in cm
5. Classes:
  - Iris Setosa
  - Iris Versicolour
  - Iris Virginica



# Getting Data

```
> iris <- read.csv("iris.data.csv", header=TRUE)

> library(datasets)

> iris

> colnames(iris) <- c("Sepal.Length", "Sepal.Width", "Petal.Length",
"Petal.Width", "Species")
```

```
> head(iris)
 Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1 5.1 3.5 1.4 0.2 setosa
2 4.9 3.0 1.4 0.2 setosa
3 4.7 3.2 1.3 0.2 setosa
4 4.6 3.1 1.5 0.2 setosa
5 5.0 3.6 1.4 0.2 setosa
6 5.4 3.9 1.7 0.4 setosa

> nrow(iris)
[1] 150

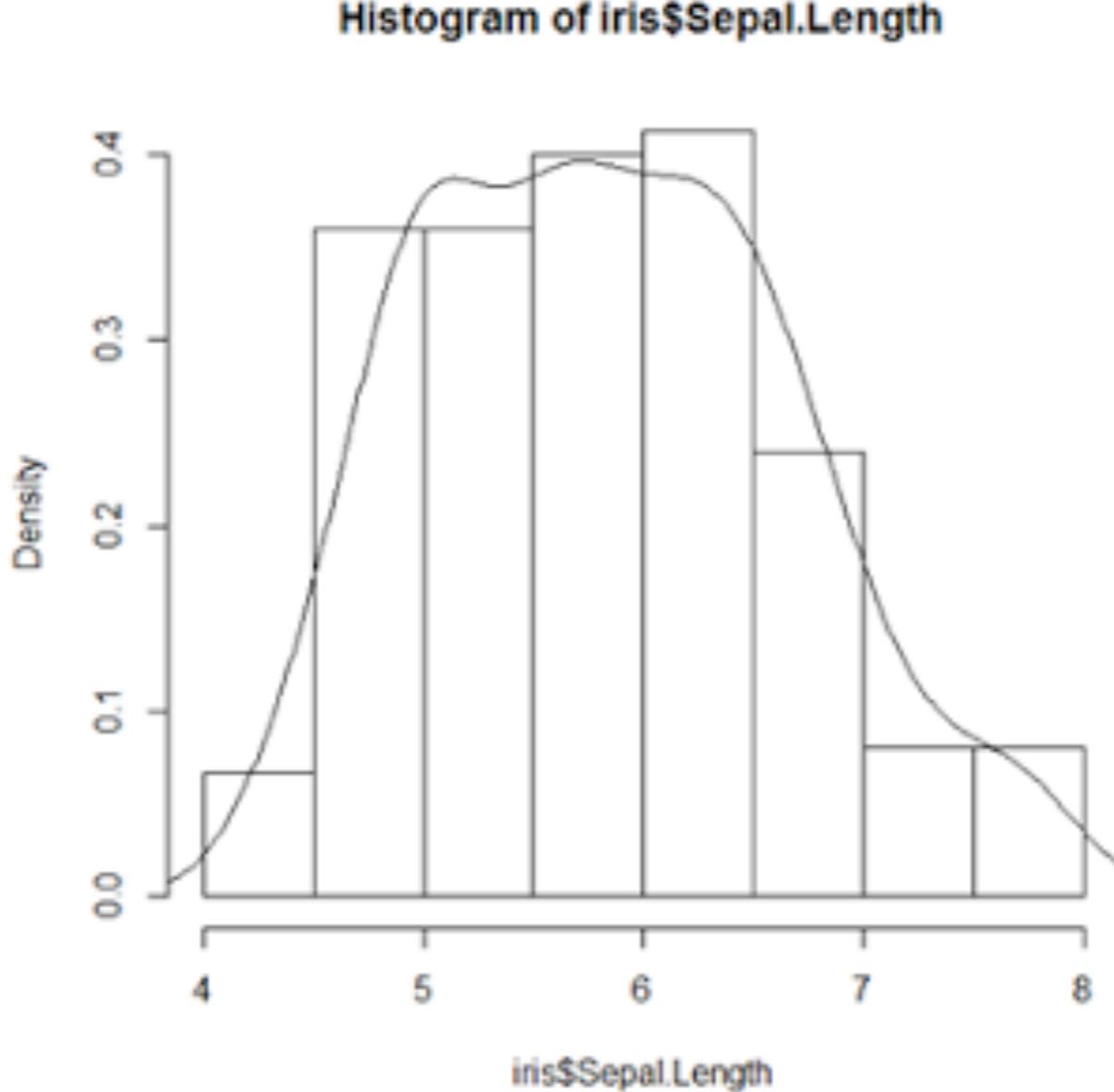
> table(iris$Species)

setosa versicolor virginica
 50 50 50
>
```

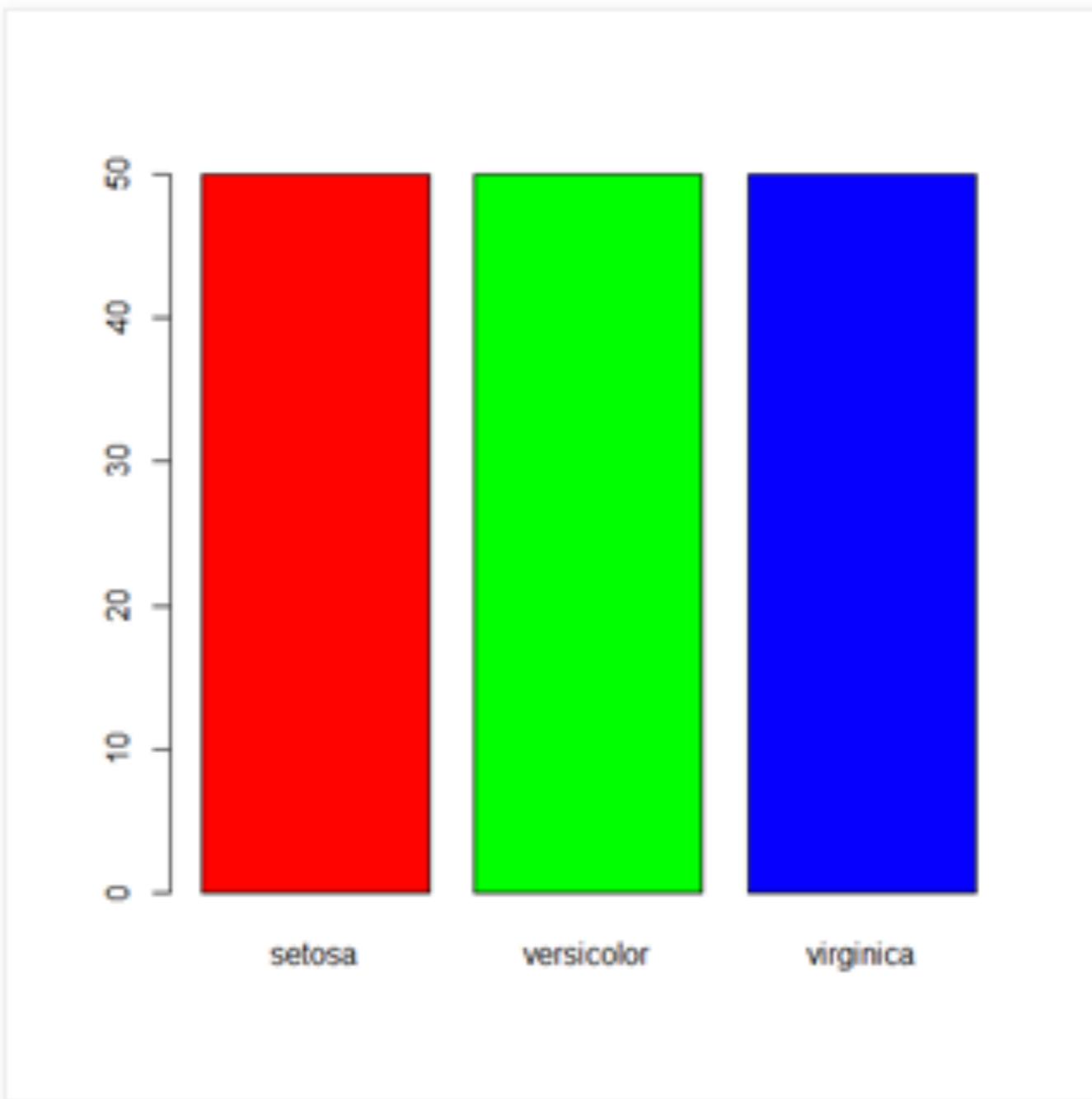
# Data Visualization

- Visualizing existing data is a very useful way to come up with ideas about what features should be included.
- "Dataframe" in R is a common way where data samples are organized in a tabular structure.

```
> # Plot the histogram
> hist(iris$Sepal.Length, breaks=10, prob=T)
> # Plot the density curve
> lines(density(iris$Sepal.Length))
>
```

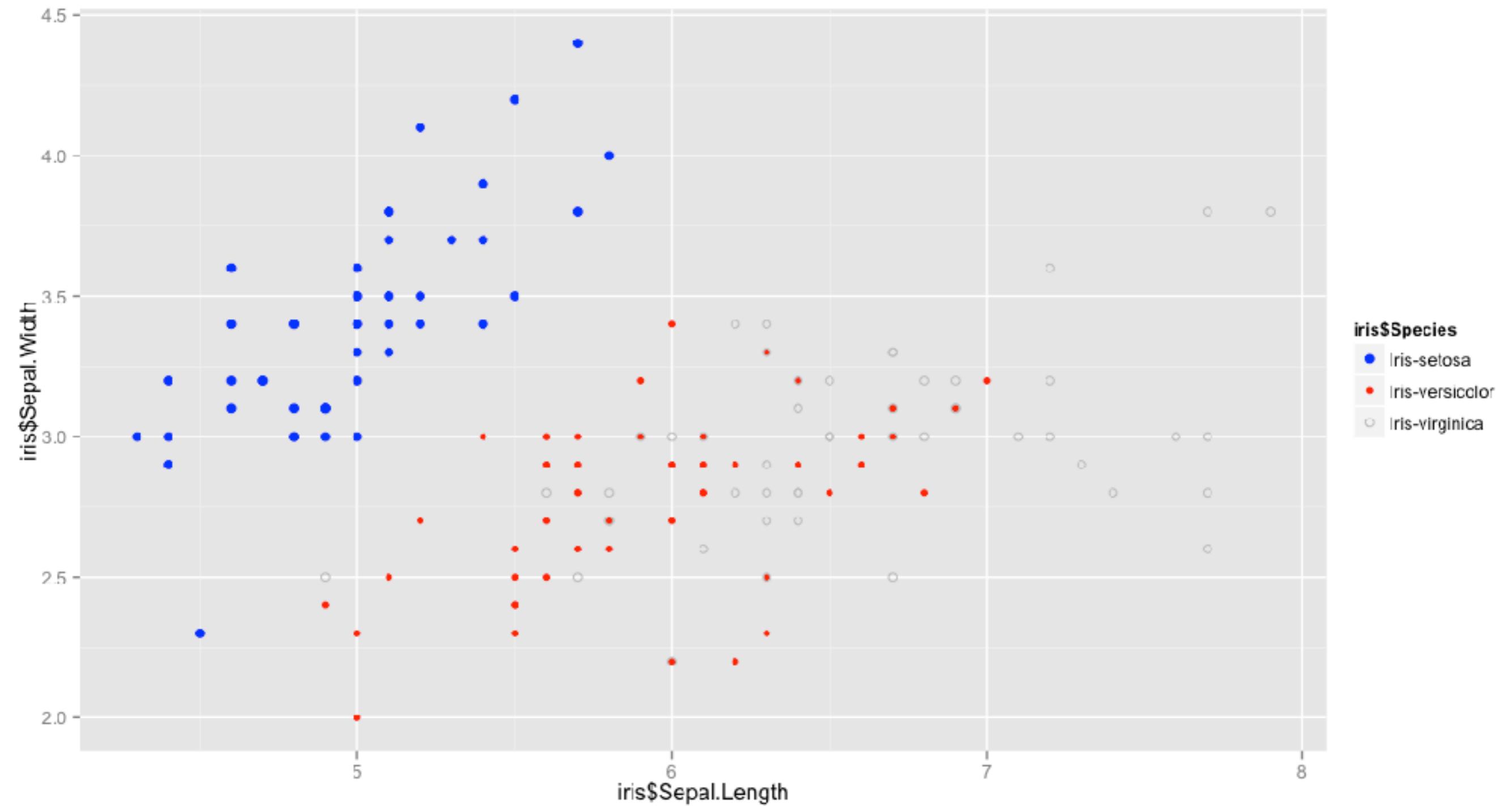


```
> categories <- table(iris$Species)
> barplot(categories, col=c('red', 'green', 'blue'))
>
```

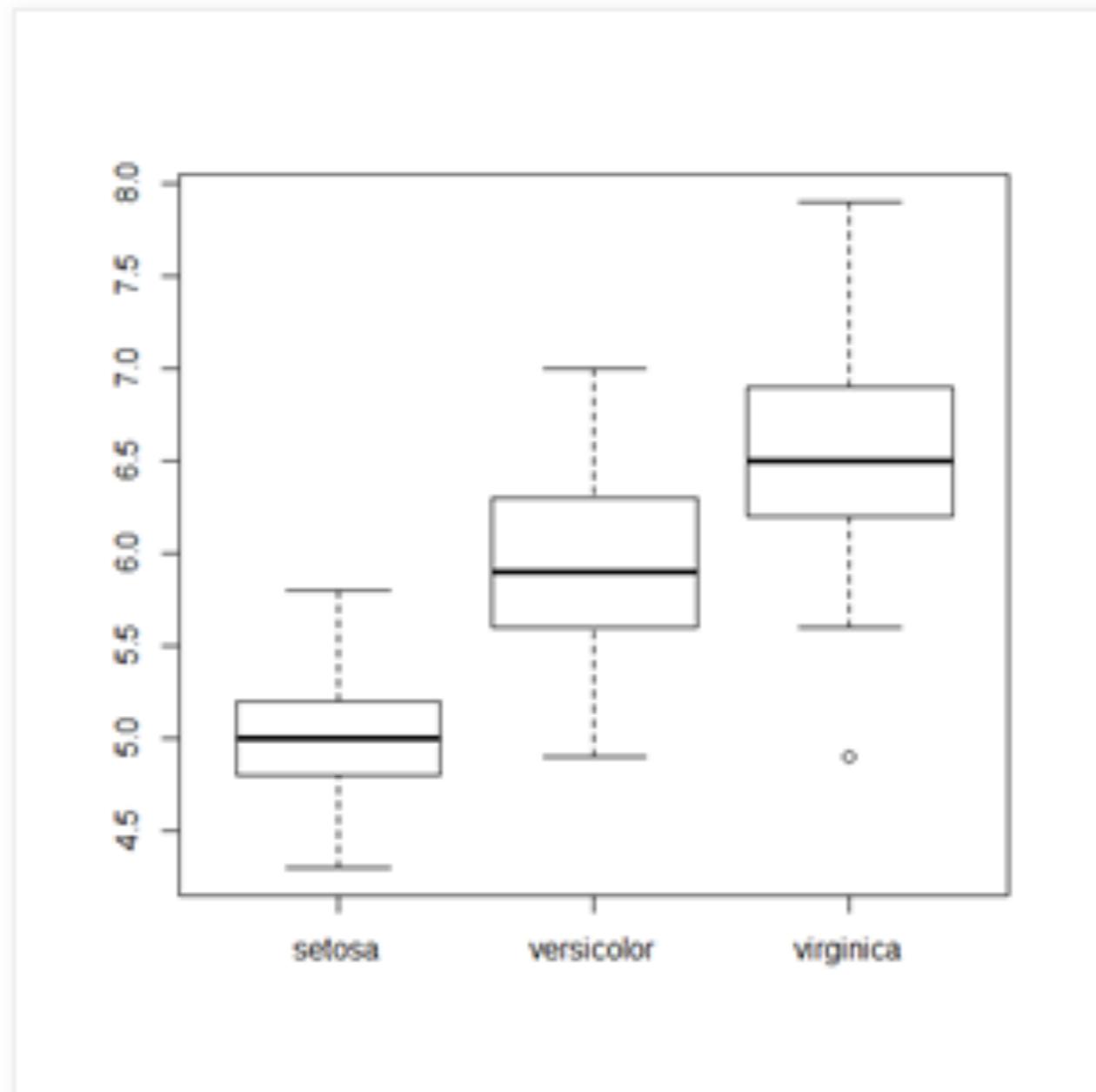




```
ggplot(iris, aes(x=iris$SepalLength, y=iris$SepalWidth,
group=iris$Species, shape=iris$Species)) +
 geom_point(aes(colour=iris$Species)) +
 scale_shape_manual(values=c(19,20,21))+
 scale_colour_manual(values=c("blue", "red","gray"))
```



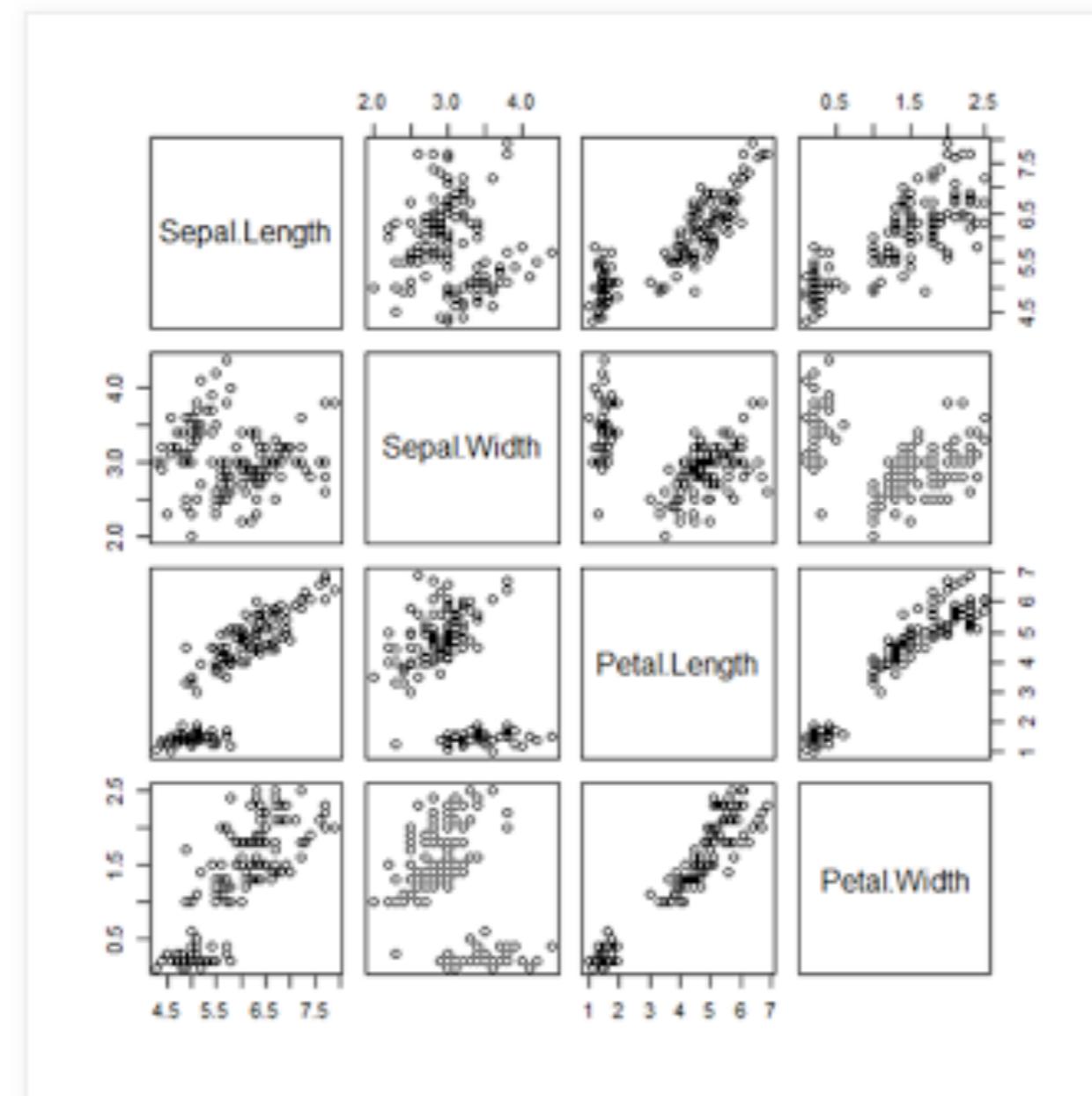
```
> boxplot(Sepal.Length~Species, data=iris)
>
```



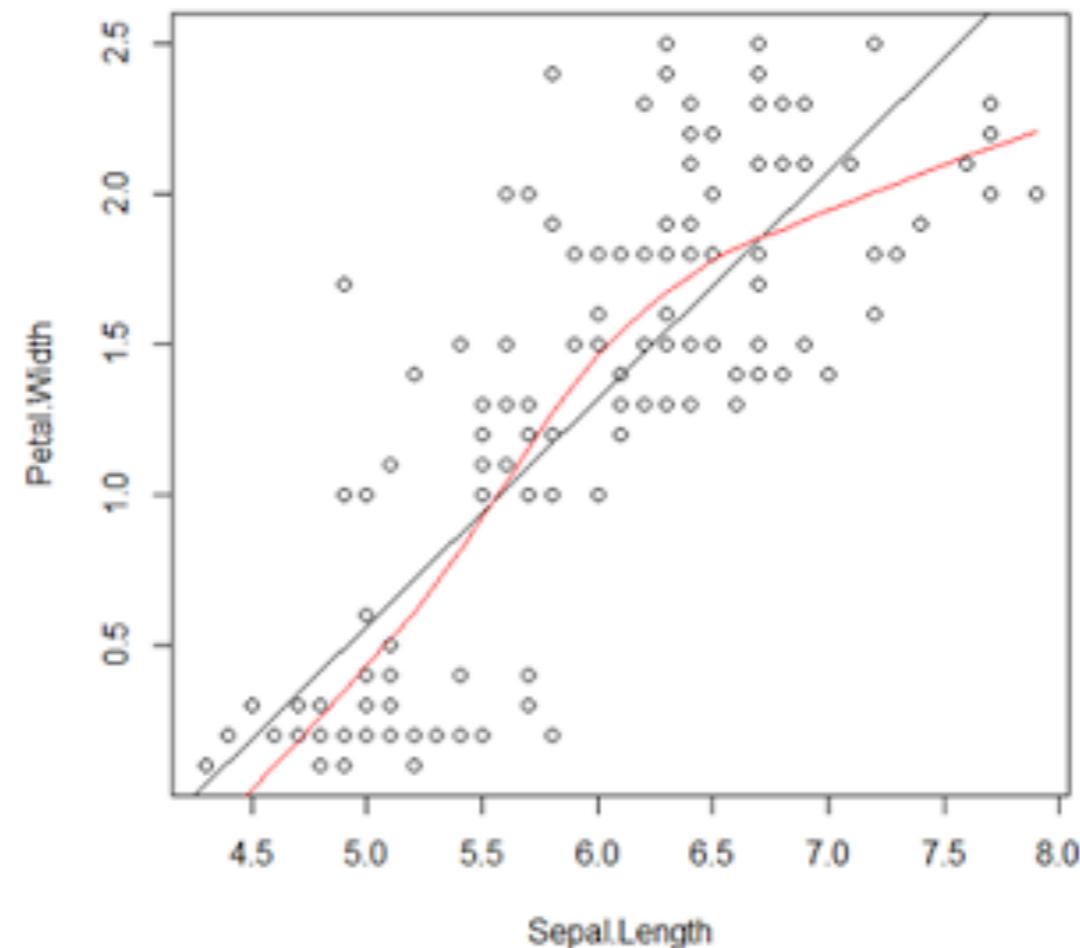
```

> # Scatter plot for all pairs
> pairs(iris[,c(1,2,3,4)])
> # Compute the correlation matrix
> cor(iris[,c(1,2,3,4)])
 Sepal.Length Sepal.Width Petal.Length Petal.Width
Sepal.Length 1.0000000 -0.1170695 0.8716902 0.8179410
Sepal.Width -0.1170695 1.0000000 -0.4284401 -0.3661259
Petal.Length 0.8716902 -0.4284401 1.0000000 0.9628654
Petal.Width 0.8179410 -0.3661259 0.9628654 1.0000000
>

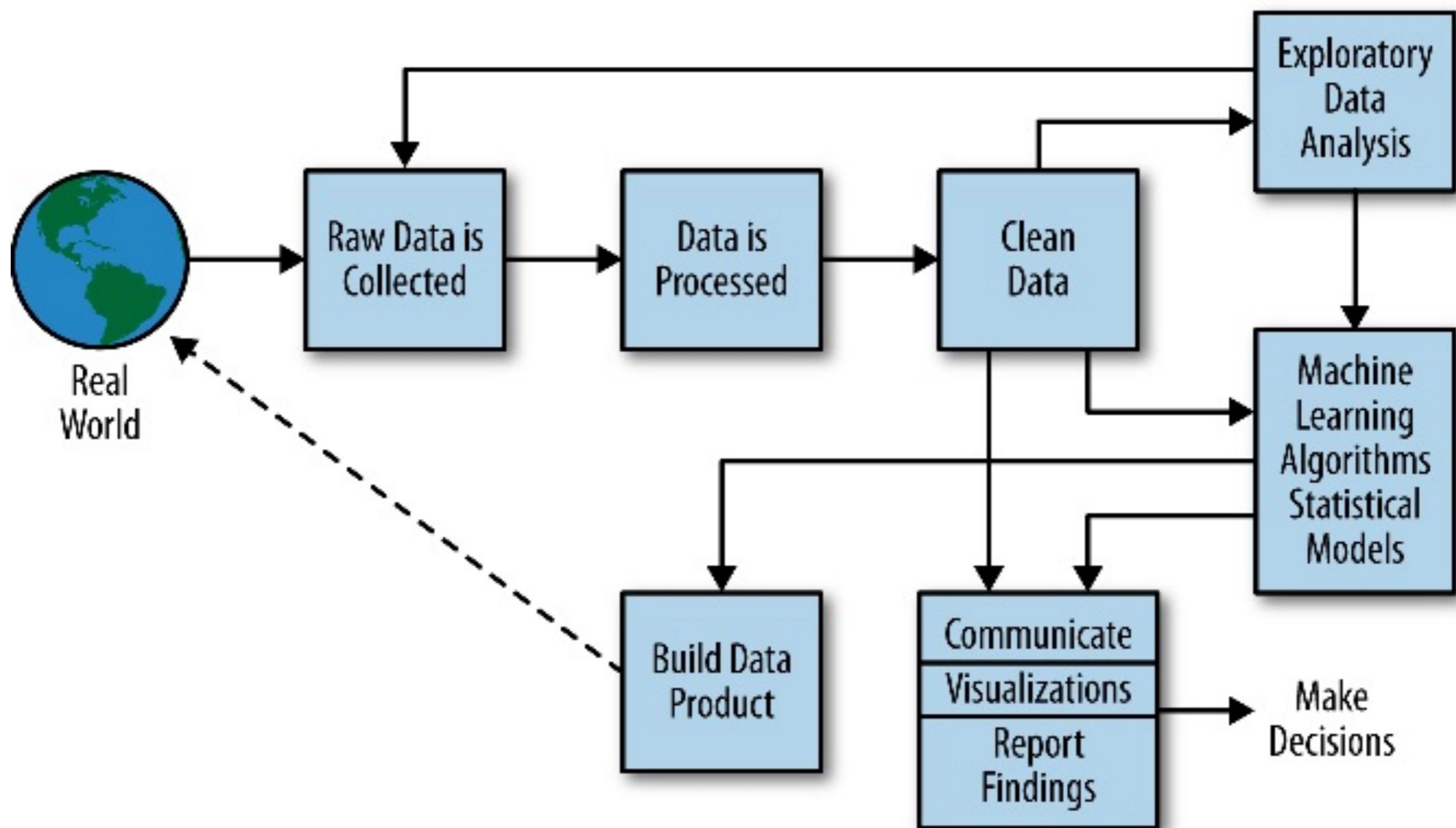
```



```
> # First plot the 2 variables
> plot(Petal.Width~Sepal.Length, data=iris)
> # Learn the regression model
> model <- lm(Petal.Width~Sepal.Length, data=iris)
> # Plot the regression line
> abline(model)
> # Now learn the local linear model
> model2 <- lowess(iris$Petal.Width~iris$Sepal.Length)
> lines(model2, col="red")
>
```



# Conclusion



# Preparing Training Data

At this step, the purpose is to transform the raw data in a form that can fit into the data mining model.

- Data sampling
- Data validation and handle missing data
- Normalize numeric value into a uniform range
- Compute aggregated value (a special case is to compute frequency counts)
- Expand categorical field to binary fields
- Discretize numeric value into categories
- Create derived fields from existing fields
- Reduce dimensionality
- Power and Log transformation

# Data Sampling

```
> # select 10 records out from iris with replacement
> index <- sample(1:nrow(iris), 10, replace=T)
> index
[1] 133 36 107 140 66 67 36 3 97 37
> irissample <- iris[index,]
> irissample
 Sepal.Length Sepal.Width Petal.Length Petal.Width Species
133 6.4 2.8 5.6 2.2 virginica
36 5.0 3.2 1.2 0.2 setosa
107 4.9 2.5 4.5 1.7 virginica
140 6.9 3.1 5.4 2.1 virginica
66 6.7 3.1 4.4 1.4 versicolor
67 5.6 3.0 4.5 1.5 versicolor
36.1 5.0 3.2 1.2 0.2 setosa
3 4.7 3.2 1.3 0.2 setosa
97 5.7 2.9 4.2 1.3 versicolor
37 5.5 3.5 1.3 0.2 setosa
>
```

# Impute missing data

- Discard the whole record
- Infer missing value based on the data of other record. Approach is to fill the missing data with the average or the median.

```
> # Create some missing data
> irissample[10, 1] <- NA
> irissample
 Sepal.Length Sepal.Width Petal.Length Petal.Width Species
133 6.4 2.8 5.6 2.2 virginica
 36 5.0 3.2 1.2 0.2 setosa
107 4.9 2.5 4.5 1.7 virginica
140 6.9 3.1 5.4 2.1 virginica
 66 6.7 3.1 4.4 1.4 versicolor
 67 5.6 3.0 4.5 1.5 versicolor
36.1 5.0 3.2 1.2 0.2 setosa
 3 4.7 3.2 1.3 0.2 setosa
 97 5.7 2.9 4.2 1.3 versicolor
 37 NA 3.5 1.3 0.2 setosa
...
```

```
> library(e1071)
Loading required package: class
Warning message:
package 'e1071' was built under R version 2.14.2
> fixIris1 <- impute(irissample[,1:4], what='mean')
> fixIris1
 Sepal.Length Sepal.Width Petal.Length Petal.Width
133 6.400000 2.8 5.6 2.2
36 5.000000 3.2 1.2 0.2
107 4.900000 2.5 4.5 1.7
140 6.900000 3.1 5.4 2.1
66 6.700000 3.1 4.4 1.4
67 5.600000 3.0 4.5 1.5
36.1 5.000000 3.2 1.2 0.2
3 4.700000 3.2 1.3 0.2
97 5.700000 2.9 4.2 1.3
37 5.655556 3.5 1.3 0.2
> fixIris2 <- impute(irissample[,1:4], what='median')
> fixIris2
 Sepal.Length Sepal.Width Petal.Length Petal.Width
133 6.4 2.8 5.6 2.2
36 5.0 3.2 1.2 0.2
107 4.9 2.5 4.5 1.7
140 6.9 3.1 5.4 2.1
66 6.7 3.1 4.4 1.4
67 5.6 3.0 4.5 1.5
36.1 5.0 3.2 1.2 0.2
3 4.7 3.2 1.3 0.2
97 5.7 2.9 4.2 1.3
37 5.6 3.5 1.3 0.2
>
```

# Normalize numeric value

```
> # scale the columns
> # x-mean(x)/standard deviation
> scaleiris <- scale(iris[, 1:4])
> head(scaleiris)
 Sepal.Length Sepal.Width Petal.Length Petal.Width
[1,] -0.8976739 1.01560199 -1.335752 -1.311052
[2,] -1.1392005 -0.13153881 -1.335752 -1.311052
[3,] -1.3807271 0.32731751 -1.392399 -1.311052
[4,] -1.5014904 0.09788935 -1.279104 -1.311052
[5,] -1.0184372 1.24503015 -1.335752 -1.311052
[6,] -0.5353840 1.93331463 -1.165809 -1.048667
>
```

# Reduce dimensionality

There are two ways to reduce the number of input attributes.

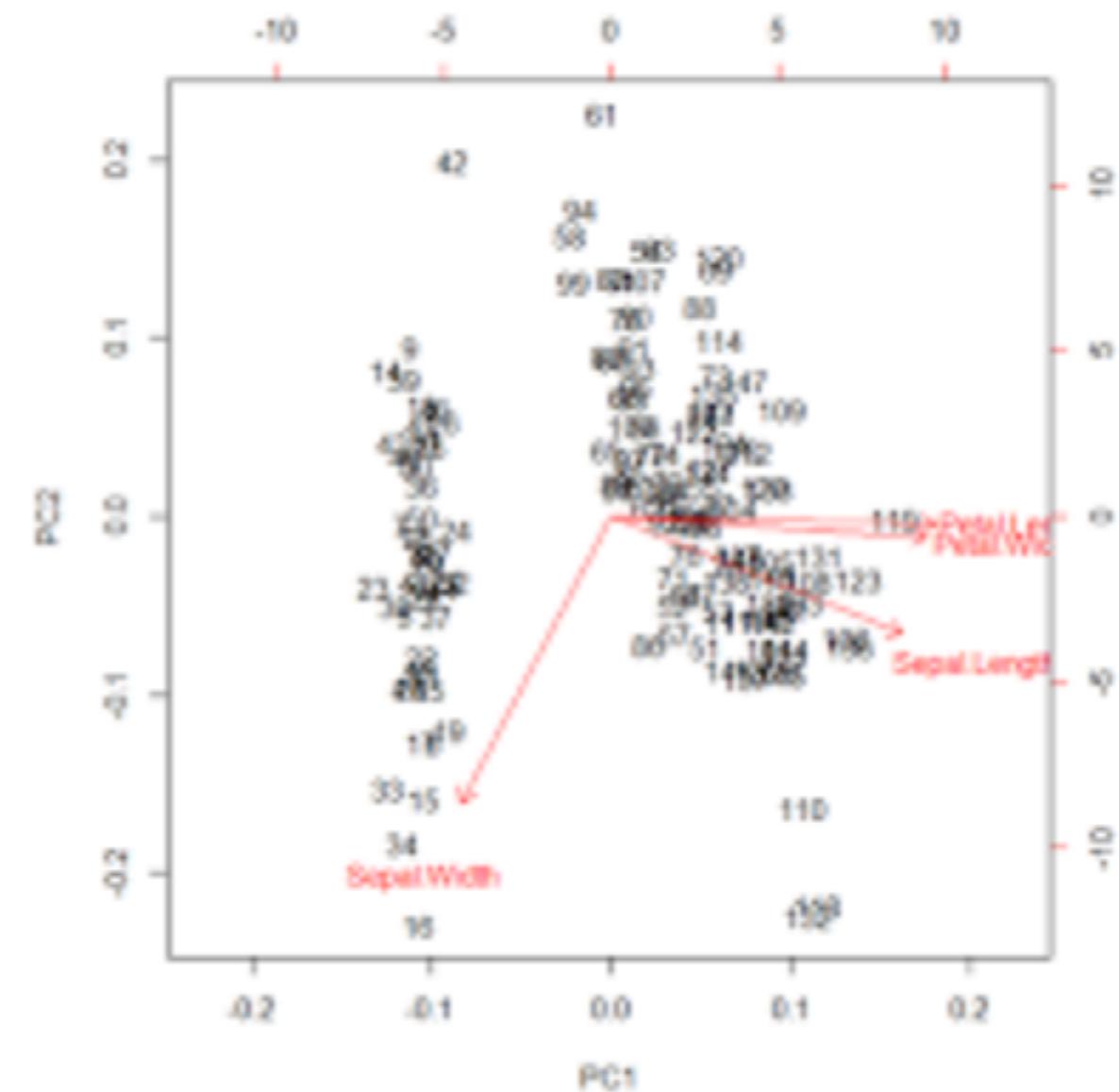
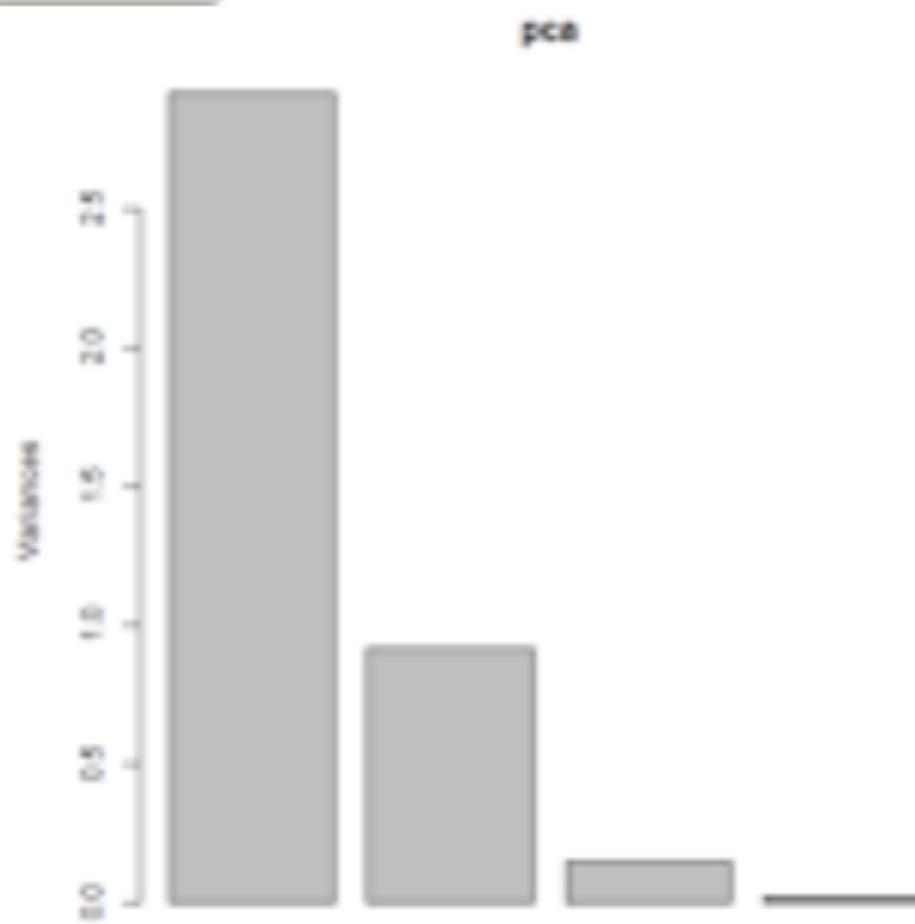
1. Removing irrelevant input variables.
2. Removing redundant input variables.

```

> # Use iris data set
> cor(iris[, -5])
 Sepal.Length Sepal.Width Petal.Length Petal.Width
Sepal.Length 1.0000000000 -0.1175697841 0.8717537759 0.8179411263
Sepal.Width -0.1175697841 1.0000000000 -0.4284401043 -0.3661259325
Petal.Length 0.8717537759 -0.4284401043 1.0000000000 0.9628654314
Petal.Width 0.8179411263 -0.3661259325 0.9628654314 1.0000000000
> # Some attributes shows high correlation, compute PCA
> pca <- prcomp(iris[, -5], scale=T)
> summary(pca)
Importance of components:
 PC1 PC2 PC3 PC4
Standard deviation 1.708361 0.9560494 0.3830886 0.1439265
Proportion of Variance 0.729620 0.2285100 0.0366900 0.0051800
Cumulative Proportion 0.729620 0.9581300 0.9948200 1.0000000
> # Notice PC1 and PC2 covers most variation
> plot(pca)
> pca$rotation
 PC1 PC2 PC3 PC4
Sepal.Length 0.5210659147 -0.37741761556 0.7195663527 0.2612862800
Sepal.Width -0.2693474425 -0.92329565954 -0.2443817795 -0.1235096196
Petal.Length 0.5804130958 -0.02449160909 -0.1421263693 -0.8014492463
Petal.Width 0.5648565358 -0.06694198697 -0.6342727371 0.5235971346
> # Project first 2 records in PCA direction
> predict(pca)[1:2,]
 PC1 PC2 PC3 PC4
[1,] -2.257141176 -0.4784238321 0.1272796237 0.02408750846
[2,] -2.074013015 0.6718826870 0.2338255167 0.10266284468
> # plot all points in top 2 PCA direction
> biplot(pca)

```

Pin it



# Add derived attributes

```
> iris2 <- transform(iris, ratio=round(Sepal.Length/Sepal.Width, 2))
> head(iris2)
 Sepal.Length Sepal.Width Petal.Length Petal.Width Species ratio
1 5.1 3.5 1.4 0.2 setosa 1.46
2 4.9 3.0 1.4 0.2 setosa 1.63
3 4.7 3.2 1.3 0.2 setosa 1.47
4 4.6 3.1 1.5 0.2 setosa 1.48
5 5.0 3.6 1.4 0.2 setosa 1.39
6 5.4 3.9 1.7 0.4 setosa 1.38
```

# Discretize numeric value into categories

```
> # Equal width cuts
> segments <- 10
> maxL <- max(iris$Petal.Length)
> minL <- min(iris$Petal.Length)
> theBreaks <- seq(minL, maxL,
+ by=(maxL-minL)/segments)
> cutPetalLength <- cut(iris$Petal.Length,
+ breaks=theBreaks,
+ include.lowest=T)
> newdata <- data.frame(orig.Petal.Len=iris$Petal.Length,
+ cut.Petal.Len=cutPetalLength)
> head(newdata)
 orig.Petal.Len cut.Petal.Len
1 1.4 [1,1.59]
2 1.4 [1,1.59]
3 1.3 [1,1.59]
4 1.5 [1,1.59]
5 1.4 [1,1.59]
6 1.7 (1.59,2.18]
>
> # Constant frequency / height
> myBreaks <- quantile(iris$Petal.Length,
+ probs=seq(0,1,1/segments))
> cutPetalLength2 <- cut(iris$Petal.Length,
+ breaks=myBreaks,
+ include.lowest=T)
> newdata2 <- data.frame(orig.Petal.Len=iris$Petal.Length,
+ cut.Petal.Len=cutPetalLength2)
> head(newdata2)
 orig.Petal.Len cut.Petal.Len
1 1.4 [1,1.4]
2 1.4 [1,1.4]
3 1.3 [1,1.4]
4 1.5 (1.4,1.5]
5 1.4 [1,1.4]
6 1.7 (1.7,3.9]
```

# Binarize categorical attributes

```
> cat <- levels(iris$Species)
> cat
[1] "setosa" "versicolor" "virginica"
> binarize <- function(x) {return(iris$Species == x)}
> newcols <- sapply(cat, binarize)
> colnames(newcols) <- cat
> data <- cbind(iris[,c('Species')], newcols)
> data[45:55,]
 setosa versicolor virginica
[1,] 1 1 0 0
[2,] 1 1 0 0
[3,] 1 1 0 0
[4,] 1 1 0 0
[5,] 1 1 0 0
[6,] 1 1 0 0
[7,] 2 0 1 0
[8,] 2 0 1 0
[9,] 2 0 1 0
[10,] 2 0 1 0
[11,] 2 0 1 0
```

# Iris Data Preparation

```
> set.seed(1234)
> ind <- sample(2, nrow(iris), replace=TRUE, prob=c(0.7, 0.3))
> trainData <- iris[ind==1,]
> testData <- iris[ind==2,]
```

# Decision Tree

## Conditional Inference Trees

### Description

Recursive partitioning for continuous, censored, ordered, nominal and multivariate response variables in a conditional inference framework.

### Usage

```
ctree(formula, data, subset = NULL, weights = NULL,
 controls = ctree_control(), xtrafo = ptrtrafo, ytrafo = ptrtrafo,
 scores = NULL)
```

### Arguments

- formula** a symbolic description of the model to be fit. Note that symbols like : and – will not work and the tree will make use of all variables listed on the rhs of **formula**.
- data** a data frame containing the variables in the model.
- subset** an optional vector specifying a subset of observations to be used in the fitting process.
- weights** an optional vector of weights to be used in the fitting process. Only non-negative integer valued weights are allowed.
- controls** an object of class [TreeControl](#), which can be obtained using [ctree\\_control](#).
- xtrafo** a function to be applied to all input variables. By default, the [ptrtrafo](#) function is applied.
- ytrafo** a function to be applied to all response variables. By default, the [ptrtrafo](#) function is applied.
- scores** an optional named list of scores to be attached to ordered factors.

# Decision Tree - Create Model

```
> library(party)
> myFormula <- Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width
> iris_ctree <- ctree(myFormula, data=trainData)
> # check the prediction
> table(predict(iris_ctree), trainData$Species)
```

|            | setosa | versicolor | virginica |
|------------|--------|------------|-----------|
| setosa     | 40     | 0          | 0         |
| versicolor | 0      | 37         | 3         |
| virginica  | 0      | 1          | 31        |

```
> print(iris_ctree)
```

Conditional inference tree with 4 terminal nodes

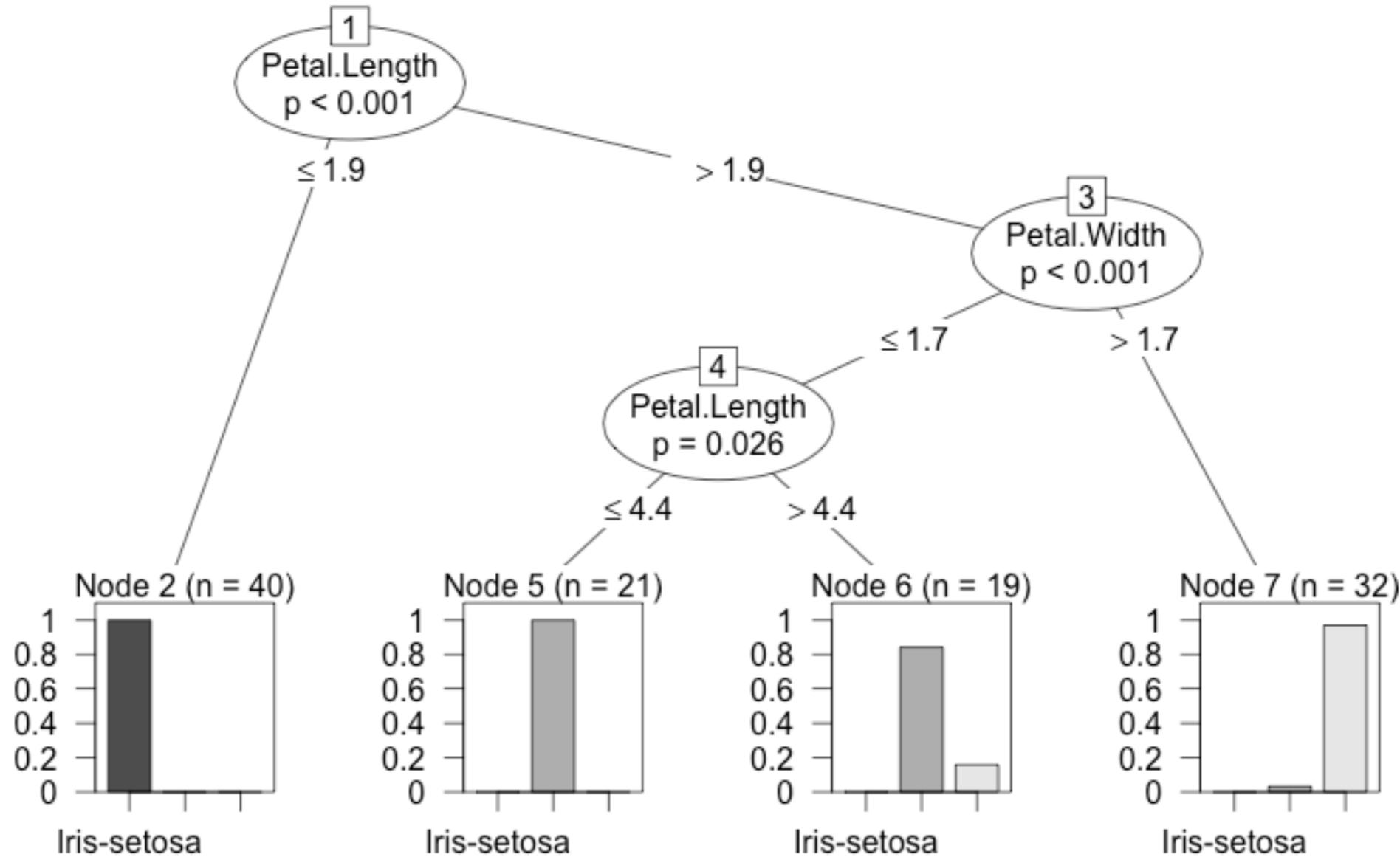
Response: Species

Inputs: Sepal.Length, Sepal.Width, Petal.Length, Petal.Width

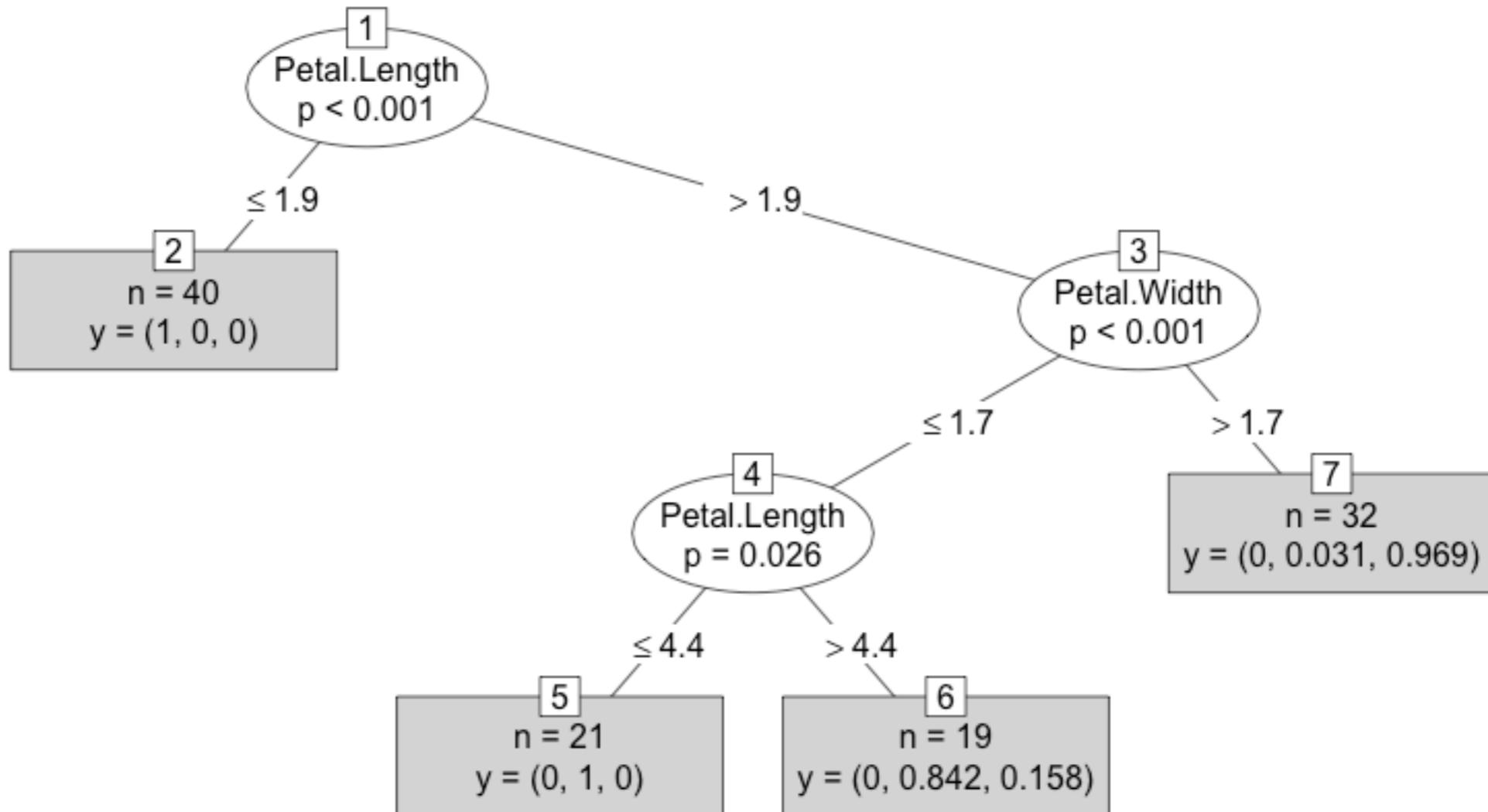
Number of observations: 112

- 1) Petal.Length <= 1.9; criterion = 1, statistic = 104.643  
2)\* weights = 40
- 1) Petal.Length > 1.9
  - 3) Petal.Width <= 1.7; criterion = 1, statistic = 48.939
    - 4) Petal.Length <= 4.4; criterion = 0.974, statistic = 7.397  
5)\* weights = 21
    - 4) Petal.Length > 4.4  
6)\* weights = 19
  - 3) Petal.Width > 1.7  
7)\* weights = 32

```
> plot(iris_ctree)
```



```
> plot(iris_ctree, type="simple")
```



# Decision Tree - Prediction

```
> # predict on test data
> testPred <- predict(iris_ctree, newdata = testData)
> table(testPred, testData$Species)
```

| testPred   | setosa | versicolor | virginica |
|------------|--------|------------|-----------|
| setosa     | 10     | 0          | 0         |
| versicolor | 0      | 12         | 2         |
| virginica  | 0      | 0          | 14        |

# Confusion Matrix

```
install.packages("caret")
library(caret)
confusionMatrix(testPred, testdata$Species)
```

| Confusion Matrix and Statistics |             |                 |                |
|---------------------------------|-------------|-----------------|----------------|
| Prediction                      | Reference   |                 |                |
|                                 | Iris-setosa | Iris-versicolor | Iris-virginica |
| Iris-setosa                     | 15          | 0               | 0              |
| Iris-versicolor                 | 0           | 13              | 2              |
| Iris-virginica                  | 0           | 0               | 13             |
| Overall Statistics              |             |                 |                |
| Accuracy : 0.9535               |             |                 |                |
| 95% CI : (0.8419, 0.9943)       |             |                 |                |
| No Information Rate : 0.3488    |             |                 |                |
| P-Value [Acc > NIR] : < 2.2e-16 |             |                 |                |
| Kappa : 0.9303                  |             |                 |                |
| McNemar's Test P-Value : NA     |             |                 |                |

### Statistics by Class:

|                      | Class: Iris-setosa | Class: Iris-versicolor | Class: Iris-virginica |
|----------------------|--------------------|------------------------|-----------------------|
| Sensitivity          | 1.0000             | 1.0000                 | 0.8667                |
| Specificity          | 1.0000             | 0.9333                 | 1.0000                |
| Pos Pred Value       | 1.0000             | 0.8667                 | 1.0000                |
| Neg Pred Value       | 1.0000             | 1.0000                 | 0.9333                |
| Prevalence           | 0.3488             | 0.3023                 | 0.3488                |
| Detection Rate       | 0.3488             | 0.3023                 | 0.3023                |
| Detection Prevalence | 0.3488             | 0.3488                 | 0.3023                |
| Balanced Accuracy    | 1.0000             | 0.9667                 | 0.9333                |

# K-NN

## k-Nearest Neighbour Classification

### Description

k-nearest neighbour classification for test set from training set. For each row of the test set, the `k` nearest (in Euclidean distance) training set vectors are found, and the classification is decided by majority vote, with ties broken at random. If there are ties for the `k`th nearest vector, all candidates are included in the vote.

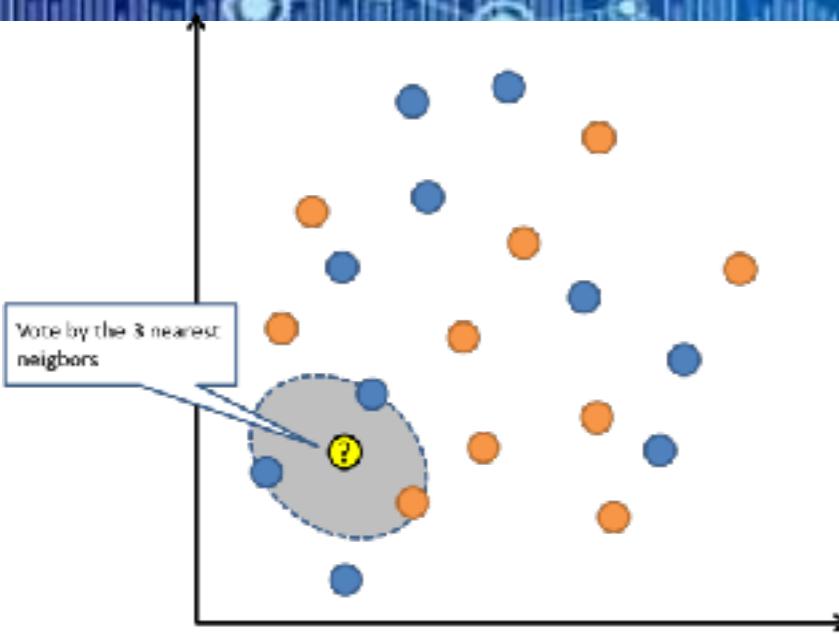
### Usage

```
knn(train, test, cl, k = 1, l = 0, prob = FALSE, use.all = TRUE)
```

### Arguments

- `train` matrix or data frame of training set cases.
- `test` matrix or data frame of test set cases. A vector will be interpreted as a row vector for a single case.
- `cl` factor of true classifications of training set
- `k` number of neighbours considered.
- `l` minimum vote for definite decision, otherwise doubt. (More precisely, less than `k`-1 dissenting votes are allowed, even if `k` is increased by ties.)
- `prob` If this is true, the proportion of the votes for the winning class are returned as attribute `prob`.
- `use.all` controls handling of ties. If true, all distances equal to the `k`th largest are included. If false, a random selection of distances equal to the `k`th is chosen to use exactly `k` neighbours.

# K-NN



```
> library(class)
> train_input <- as.matrix(iristrain[,-5])
> train_output <- as.vector(iristrain[,5])
> test_input <- as.matrix(iristest[,-5])
> prediction <- knn(train_input, test_input,
+ train_output, k=5)
> table(prediction, iristest$Species)

prediction setosa versicolor virginica
 setosa 10 0 0
 versicolor 0 10 1
 virginica 0 0 9
>
```

```
> confusionMatrix(prediction, testdata$Species)
```

Confusion Matrix and Statistics

| Prediction      | Reference   |                 |                |
|-----------------|-------------|-----------------|----------------|
|                 | Iris-setosa | Iris-versicolor | Iris-virginica |
| Iris-setosa     | 15          | 0               | 0              |
| Iris-versicolor | 0           | 12              | 0              |
| Iris-virginica  | 0           | 1               | 15             |

Overall Statistics

Accuracy : 0.9767

95% CI : (0.8771, 0.9994)

No Information Rate : 0.3488

P-Value [Acc > NIR] : < 2.2e-16

Kappa : 0.965

McNemar's Test P-Value : NA

Statistics by Class:

Class: Iris-setosa Class: Iris-versicolor

|                      |        |        |
|----------------------|--------|--------|
| Sensitivity          | 1.0000 | 0.9231 |
| Specificity          | 1.0000 | 1.0000 |
| Pos Pred Value       | 1.0000 | 1.0000 |
| Neg Pred Value       | 1.0000 | 0.9677 |
| Prevalence           | 0.3488 | 0.3023 |
| Detection Rate       | 0.3488 | 0.2791 |
| Detection Prevalence | 0.3488 | 0.2791 |
| Balanced Accuracy    | 1.0000 | 0.9615 |

Class: Iris-virginica

|                      |        |
|----------------------|--------|
| Sensitivity          | 1.0000 |
| Specificity          | 0.9643 |
| Pos Pred Value       | 0.9375 |
| Neg Pred Value       | 1.0000 |
| Prevalence           | 0.3488 |
| Detection Rate       | 0.3488 |
| Detection Prevalence | 0.3721 |
| Balanced Accuracy    | 0.9821 |



# Support Vector Machine

svm {e1071}

R Documentation

## Support Vector Machines

### Description

`svm` is used to train a support vector machine. It can be used to carry out general regression and classification (of nu and epsilon-type), as well as density-estimation. A formula interface is provided.

### Usage

```
S3 method for class 'formula'
svm(formula, data = NULL, ..., subset, na.action =
na.omit, scale = TRUE)
Default S3 method:
svm(x, y = NULL, scale = TRUE, type = NULL, kernel =
"radial", degree = 3, gamma = if (is.vector(x)) 1 else 1 / ncol(x),
coef0 = 0, cost = 1, nu = 0.5,
class.weights = NULL, cachesize = 40, tolerance = 0.001, epsilon = 0.1,
shrinking = TRUE, cross = 0, probability = FALSE, fitted = TRUE,
..., subset, na.action = na.omit)
```

### Arguments

#### formula

a symbolic description of the model to be fit.

#### data

an optional data frame containing the variables in the model. By default the variables are taken from the environment in which 'svm' is called from.

#### x

a data matrix, a vector, or a sparse matrix (object of class [Matrix](#) provided by the [Matrix](#) package, or of class [matrix.csr](#) provided by the [SparseM](#) package, or of class [simple\\_triplet\\_matrix](#) provided by the [slam](#) package).

#### y

a response vector with one label for each row/component of x. Can be either a factor (for classification tasks) or a numeric vector (for regression).

# Support Vector Machine

```
> library(e1071)
> tune <- tune.svm(Species~., data=iristrain, gamma=10^(-6:-1), cost=10^(1:4))
> summary(tune)
Parameter tuning of 'svm':
- sampling method: 10-fold cross validation
- best parameters:
 gamma cost
 0.001 10000
- best performance: 0.03333333
> model <- svm(Species~., data=iristrain, method="C-classification",
kernel="radial", probability=T, gamma=0.001, cost=10000)
> prediction <- predict(model, iristest, probability=T)
> table(iristest$Species, prediction)

 prediction
 setosa versicolor virginica
setosa 10 0 0
versicolor 0 10 0
virginica 0 3 7
>
```

# Naive Bayes

## Naive Bayes Classifier

### Description

Computes the conditional a-posterior probabilities of a categorical class variable given independent predictor variables using the Bayes rule.

### Usage

```
S3 method for class 'formula'
naiveBayes(formula, data, laplace = 0, ..., subset, na.action = na.pass)
Default S3 method:
naiveBayes(x, y, laplace = 0, ...)

S3 method for class 'naiveBayes'
predict(object, newdata,
 type = c("class", "raw"), threshold = 0.001, eps = 0, ...)
```

# Naive Bayes

## Arguments

|                        |                                                                                                                                                                                                                                                                                                                                             |
|------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>x</code>         | A numeric matrix, or a data frame of categorical and/or numeric variables.                                                                                                                                                                                                                                                                  |
| <code>y</code>         | Class vector.                                                                                                                                                                                                                                                                                                                               |
| <code>formula</code>   | A formula of the form <code>class ~ x1 + x2 + ...</code> . Interactions are not allowed.                                                                                                                                                                                                                                                    |
| <code>data</code>      | Either a data frame of predictors (categorical and/or numeric) or a contingency table.                                                                                                                                                                                                                                                      |
| <code>laplace</code>   | positive double controlling Laplace smoothing. The default (0) disables Laplace smoothing.                                                                                                                                                                                                                                                  |
| <code>...</code>       | Currently not used.                                                                                                                                                                                                                                                                                                                         |
| <code>subset</code>    | For data given in a data frame, an index vector specifying the cases to be used in the training sample.<br>(NOTE: If given, this argument must be named.)                                                                                                                                                                                   |
| <code>na.action</code> | A function to specify the action to be taken if <code>NAs</code> are found. The default action is not to count them for the computation of the probability factors. An alternative is <code>na.omit</code> , which leads to rejection of cases with missing values on any required variable. (NOTE: If given, this argument must be named.) |
| <code>object</code>    | An object of class " <code>naiveBayes</code> ".                                                                                                                                                                                                                                                                                             |
| <code>newdata</code>   | A data frame with new predictors (with possibly fewer columns than the training data). Note that the column names of <code>newdata</code> are matched against the training data ones.                                                                                                                                                       |
| <code>type</code>      | If "raw", the conditional a-posterior probabilities for each class are returned, and the class with maximal probability else.                                                                                                                                                                                                               |
| <code>threshold</code> | Value replacing cells with probabilities within <code>eps</code> range.                                                                                                                                                                                                                                                                     |
| <code>eps</code>       | double for specifying an epsilon-range to apply laplace smoothing (to replace zero or close-zero probabilities by <code>threshold</code> .)                                                                                                                                                                                                 |

# Naive Bayes

```
> library(e1071)
> # Can handle both categorical and numeric input,
> # but output must be categorical
> model <- naiveBayes(Species~., data=iristrain)
> prediction <- predict(model, iristest[,-5])
> table(prediction, iristest[,5])
```

| prediction | setosa | versicolor | virginica |
|------------|--------|------------|-----------|
| setosa     | 10     | 0          | 0         |
| versicolor | 0      | 10         | 2         |
| virginica  | 0      | 0          | 8         |

```
> confusionMatrix(prediction, testdata$Species)
```

## Confusion Matrix and Statistics

|                 |  | Reference   |                 |                |
|-----------------|--|-------------|-----------------|----------------|
|                 |  | Iris-setosa | Iris-versicolor | Iris-virginica |
| Prediction      |  |             |                 |                |
| Iris-setosa     |  | 15          | 0               | 0              |
| Iris-versicolor |  | 0           | 13              | 2              |
| Iris-virginica  |  | 0           | 0               | 13             |

## Overall Statistics

Accuracy : 0.9535

95% CI : (0.8419, 0.9943)

No Information Rate : 0.3488

P-Value [Acc > NIR] : < 2.2e-16

Kappa : 0.9303

McNemar's Test P-Value : NA

## Statistics by Class:

|  | Class: Iris-setosa | Class: Iris-versicolor | Class: Iris-virginica |
|--|--------------------|------------------------|-----------------------|
|--|--------------------|------------------------|-----------------------|

|                      |        |        |        |
|----------------------|--------|--------|--------|
| Sensitivity          | 1.0000 | 1.0000 | 0.8667 |
| Specificity          | 1.0000 | 0.9333 | 1.0000 |
| Pos Pred Value       | 1.0000 | 0.8667 | 1.0000 |
| Neg Pred Value       | 1.0000 | 1.0000 | 0.9333 |
| Prevalence           | 0.3488 | 0.3023 | 0.3488 |
| Detection Rate       | 0.3488 | 0.3023 | 0.3023 |
| Detection Prevalence | 0.3488 | 0.3488 | 0.3023 |
| Balanced Accuracy    | 1.0000 | 0.9667 | 0.9333 |

# Neural Network

## Training of neural networks

### Description

`neuralnet` is used to train neural networks using backpropagation, resilient backpropagation (RPROP) with (Riedmiller, 1994) or without weight backtracking (Riedmiller and Braun, 1993) or the modified globally convergent version (GRPROP) by Anastasiadis et al. (2005). The function allows flexible settings through custom-choice of error and activation function. Furthermore the calculation of generalized weights (Intrator O. and Intrator N., 1993) is implemented.

### Usage

```
neuralnet(formula, data, hidden = 1, threshold = 0.01,
 stepmax = 1e+05, rep = 1, startweights = NULL,
 learningrate.limit = NULL,
 learningrate.factor = list(minus = 0.5, plus = 1.2),
 learningrate=NULL, lifesign = "none",
 lifesign.step = 1000, algorithm = "rprop+",
 err.fct = "sse", act.fct = "logistic",
 linear.output = TRUE, exclude = NULL,
 constant.weights = NULL, likelihood = FALSE)
```

# Neural Network

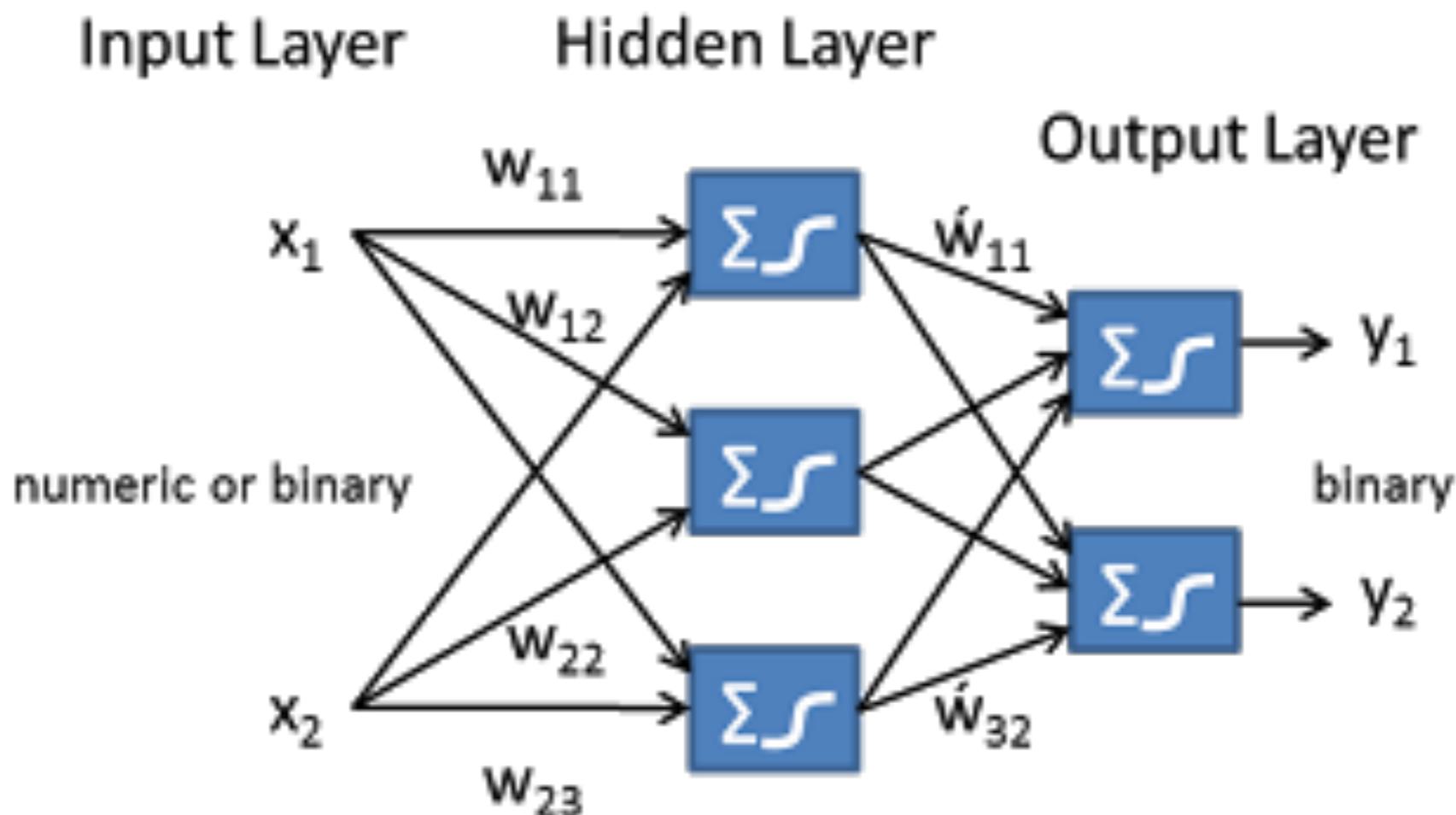
## Arguments

|                                  |                                                                                                                                           |
|----------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------|
| <code>formula</code>             | a symbolic description of the model to be fitted.                                                                                         |
| <code>data</code>                | a data frame containing the variables specified in <code>formula</code> .                                                                 |
| <code>hidden</code>              | a vector of integers specifying the number of hidden neurons (vertices) in each layer.                                                    |
| <code>threshold</code>           | a numeric value specifying the threshold for the partial derivatives of the error function as stopping criteria.                          |
| <code>stepmax</code>             | the maximum steps for the training of the neural network. Reaching this maximum leads to a stop of the neural network's training process. |
| <code>rep</code>                 | the number of repetitions for the neural network's training.                                                                              |
| <code>startweights</code>        | a vector containing starting values for the weights. The weights will not be randomly initialized.                                        |
| <code>learningrate.limit</code>  | a vector or a list containing the lowest and highest limit for the learning rate. Used only for RPROP and GRPROP.                         |
| <code>learningrate.factor</code> | a vector or a list containing the multiplication factors for the upper and lower learning rate. Used only for RPROP and GRPROP.           |
| <code>learningrate</code>        | a numeric value specifying the learning rate used by traditional backpropagation. Used only for traditional backpropagation.              |
| <code>lifesign</code>            | a string specifying how much the function will print during the calculation of the neural network. 'none', 'minimal' or 'full'.           |

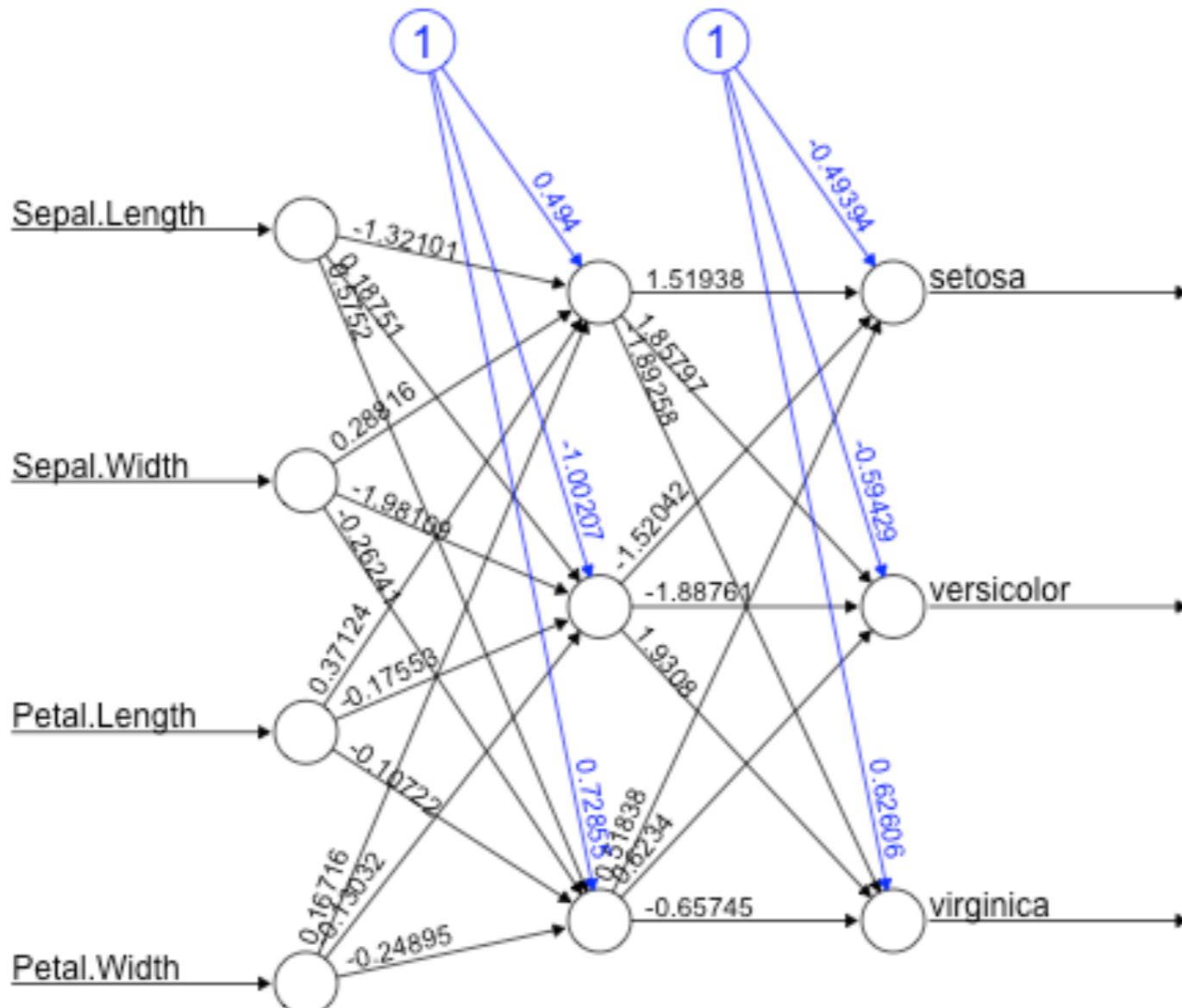
# Neural Network

|                               |                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|-------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>lifesign.step</code>    | an integer specifying the stepsize to print the minimal threshold in full lifesign mode.                                                                                                                                                                                                                                                                                                                                             |
| <code>algorithm</code>        | a string containing the algorithm type to calculate the neural network. The following types are possible: 'backprop', 'rprop+', 'rprop-', 'sag', or 'slr'. 'backprop' refers to backpropagation, 'rprop+' and 'rprop-' refer to the resilient backpropagation with and without weight backtracking, while 'sag' and 'slr' induce the usage of the modified globally convergent algorithm (grprop). See Details for more information. |
| <code>err.fct</code>          | a differentiable function that is used for the calculation of the error. Alternatively, the strings 'sse' and 'ce' which stand for the sum of squared errors and the cross-entropy can be used.                                                                                                                                                                                                                                      |
| <code>act.fct</code>          | a differentiable function that is used for smoothing the result of the cross product of the covariate or neurons and the weights. Additionally the strings, 'logistic' and 'tanh' are possible for the logistic function and tangent hyperbolicus.                                                                                                                                                                                   |
| <code>linear.output</code>    | logical. If <code>act.fct</code> should not be applied to the output neurons set <code>linear.output</code> to TRUE, otherwise to FALSE.                                                                                                                                                                                                                                                                                             |
| <code>exclude</code>          | a vector or a matrix specifying the weights, that are excluded from the calculation. If given as a vector, the exact positions of the weights must be known. A matrix with n-rows and 3 columns will exclude n weights, where the first column stands for the layer, the second column for the input neuron and the third column for the output neuron of the weight.                                                                |
| <code>constant.weights</code> | a vector specifying the values of the weights that are excluded from the training process and treated as fix.                                                                                                                                                                                                                                                                                                                        |
| <code>likelihood</code>       | logical. If the error function is equal to the negative log-likelihood function, the information criteria AIC and BIC will be calculated. Furthermore the usage of <code>confidence.interval</code> is meaningful.                                                                                                                                                                                                                   |

# Neural Network



```
2
3 library(neuralnet)
4 iris <- read.csv("iris.data.csv", header=TRUE)
5 # Prepare iris
6 set.seed(567)
7 ind <- sample(2, nrow(iris), replace=TRUE, prob=c(0.7, 0.3))
8 trainData <- iris[ind==1,]
9 testData <- iris[ind==2,]
10 nnet_iristrain <- trainData
11
12 #Binarize the categorical output
13 nnet_iristrain <- cbind(nnet_iristrain, trainData$Species == 'Iris-setosa')
14 nnet_iristrain <- cbind(nnet_iristrain, trainData$Species == 'Iris-versicolor')
15 nnet_iristrain <- cbind(nnet_iristrain, trainData$Species == 'Iris-virginica')
16 names(nnet_iristrain)[6] <- 'Setosa'
17 names(nnet_iristrain)[7] <- 'Versicolor'
18 names(nnet_iristrain)[8] <- 'Virginica'
19
20 nn <- neuralnet(Setosa+Versicolor+Virginica ~ Sepal.Length+Sepal.Width+Petal.Length+Petal.Width,
21 data=nnet_iristrain, hidden=c(3))
22
23 plot(nn)
24 mypredict <- compute(nn, testData[-5])$net.result
25 # Put multiple binary output to categorical output
26 maxidx <- function(arr) {
27 return(which(arr == max(arr)))
28 }
29 idx <- apply(mypredict, c(1), maxidx)
30 prediction <- c('Iris-setosa', 'Iris-versicolor', 'Iris-virginica')[idx]
31 table(prediction, testData$Species)
32
33
```



Error: 0.001277 Steps: 281

```
> table(prediction, testData$Species)
```

| <b>prediction</b>      | <b>Iris-setosa</b> | <b>Iris-versicolor</b> | <b>Iris-virginica</b> |
|------------------------|--------------------|------------------------|-----------------------|
| <b>Iris-setosa</b>     | 15                 | 0                      | 0                     |
| <b>Iris-versicolor</b> | 0                  | 13                     | 0                     |
| <b>Iris-virginica</b>  | 0                  | 0                      | 15                    |

>

# Clustering

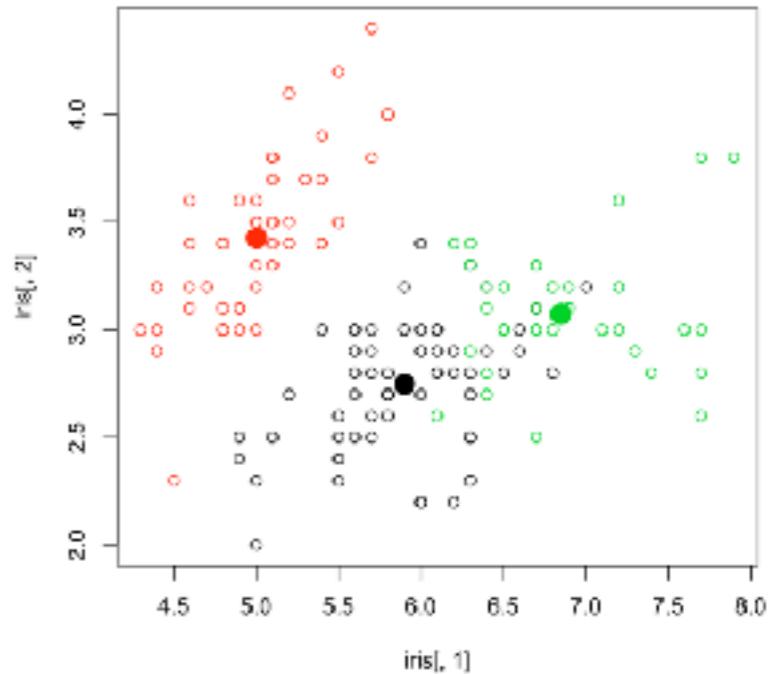
K-Means Clustering

Hierarchical Clustering

# K-Means Clustering

1. Pick an initial set of K centroids (this can be random or any other means)
2. For each data point, assign it to the member of the closest centroid according to the given distance function
3. Adjust the centroid position as the mean of all its assigned member data points. Go back to (2) until the membership isn't change and centroid position is stable.
4. Output the centroids.

```
library(stats)
set.seed(101)
km <- kmeans(iris[,1:4], 3)
plot(iris[,1], iris[,2], col=km$cluster)
points(km$centers[,c(1,2)], col=1:3, pch=19, cex=2)
```



```
table(km$cluster, iris$Species)
```

|   | setosa | versicolor | virginica |
|---|--------|------------|-----------|
| 1 | 0      | 48         | 14        |
| 2 | 50     | 0          | 0         |
| 3 | 0      | 2          | 36        |
|   |        |            |           |

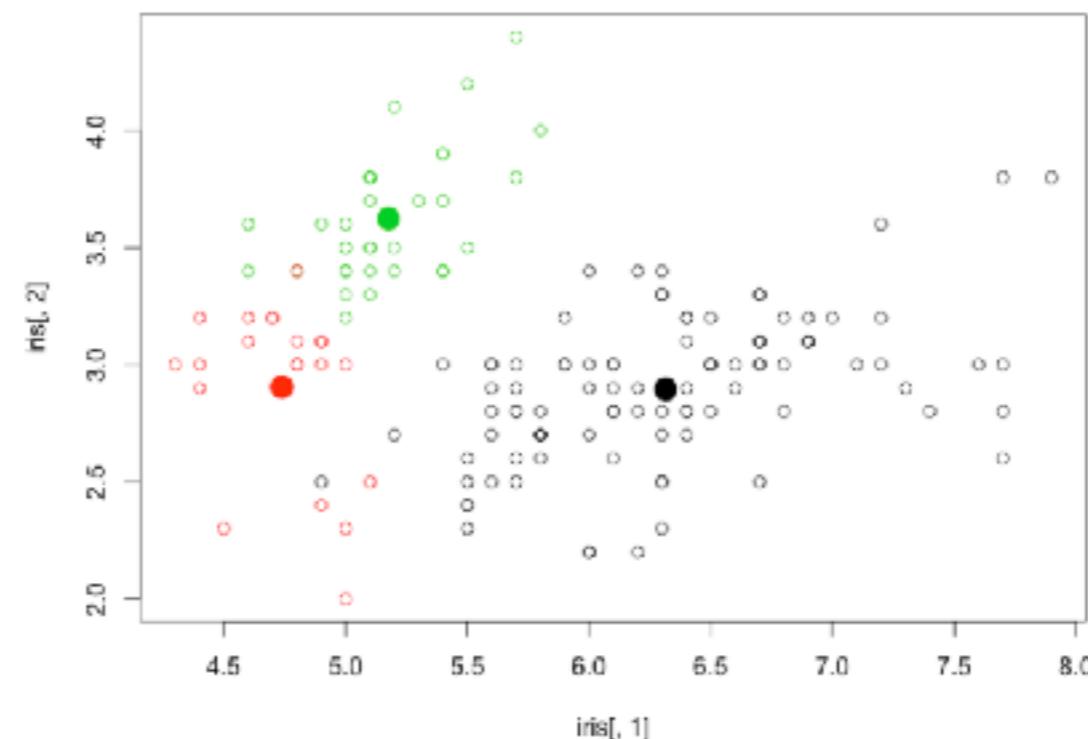
# Another round

```
set.seed(900)
```

```
km <- kmeans(iris[,1:4], 3)
```

```
plot(iris[,1], iris[,2], col=km$cluster)
```

```
points(km$centers[,c(1,2)], col=1:3, pch=19, cex=2)
```



```
table(km$cluster, iris$Species)
```

|   | setosa | versicolor | virginica |
|---|--------|------------|-----------|
| 1 | 0      | 46         | 50        |
| 2 | 17     | 4          | 0         |
| 3 | 33     | 0          | 0         |
|   |        |            |           |

# Hierarchical Clustering

Compute distance between every pairs of point/cluster.

- (a) Distance between point is just using the distance function.
- (b) Compute distance between pointA to clusterB may involve many choices (such as the min/max/avg distance between the pointA and points in the clusterB).
- (c) Compute distance between clusterA to clusterB may first compute distance of all points pairs (one from clusterA and the other from clusterB) and then pick either min/max/avg of these pairs.

Combine the two closest point/cluster into a cluster. Go back to (1) until only one big cluster remains

```
set.seed(101)

sampleiris <- iris[sample(1:150, 40),] # get samples from iris dataset

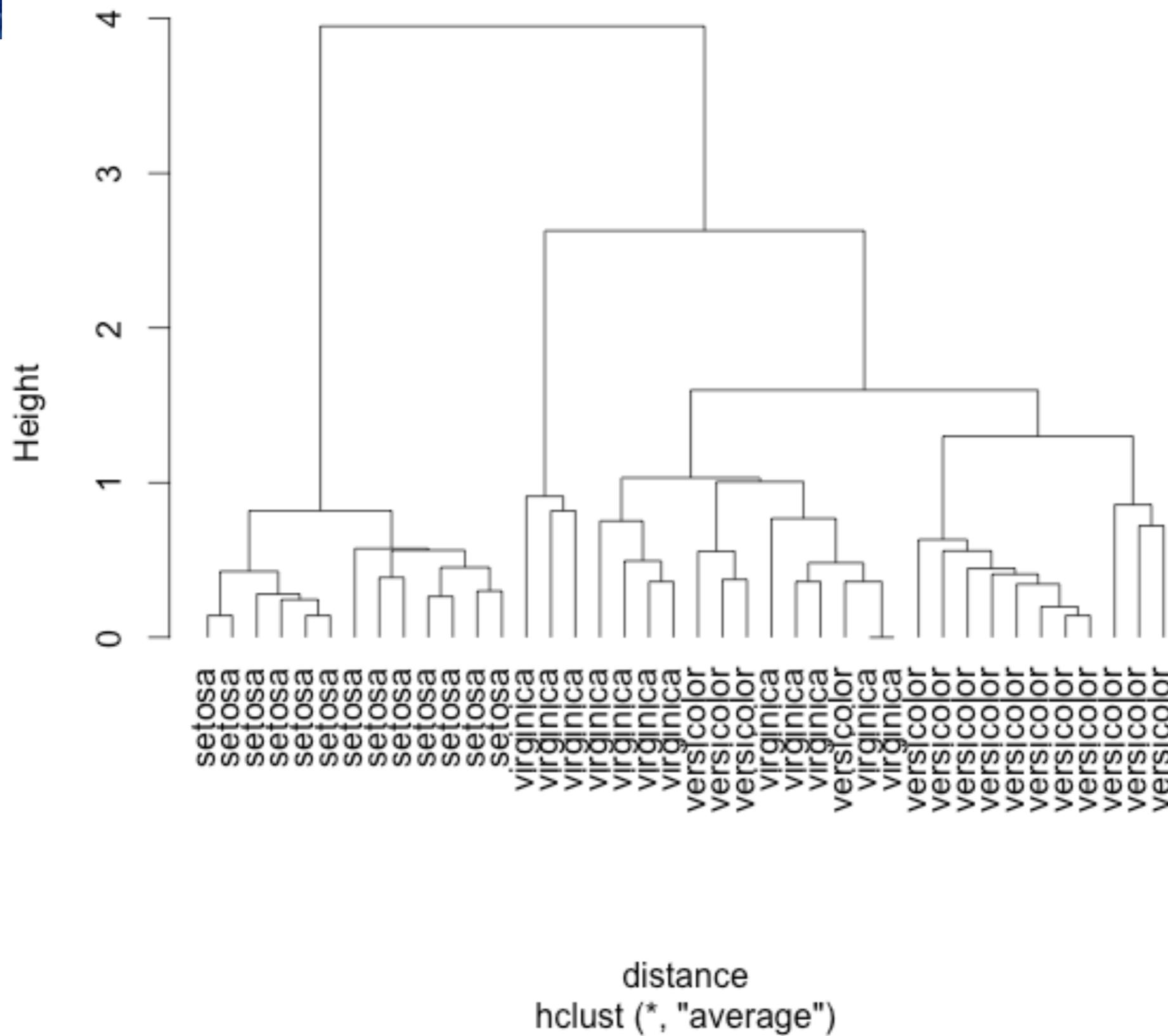
each observation has 4 variables, ie, they are interpreted as 4-D points

distance <- dist(sampleiris[,-5], method="euclidean")

cluster <- hclust(distance, method="average")

plot(cluster, hang=-1, label=sampleiris$Species)
```

## Cluster Dendrogram



# It's possible to prune the result tree.

```
par(mfrow=c(1,2))

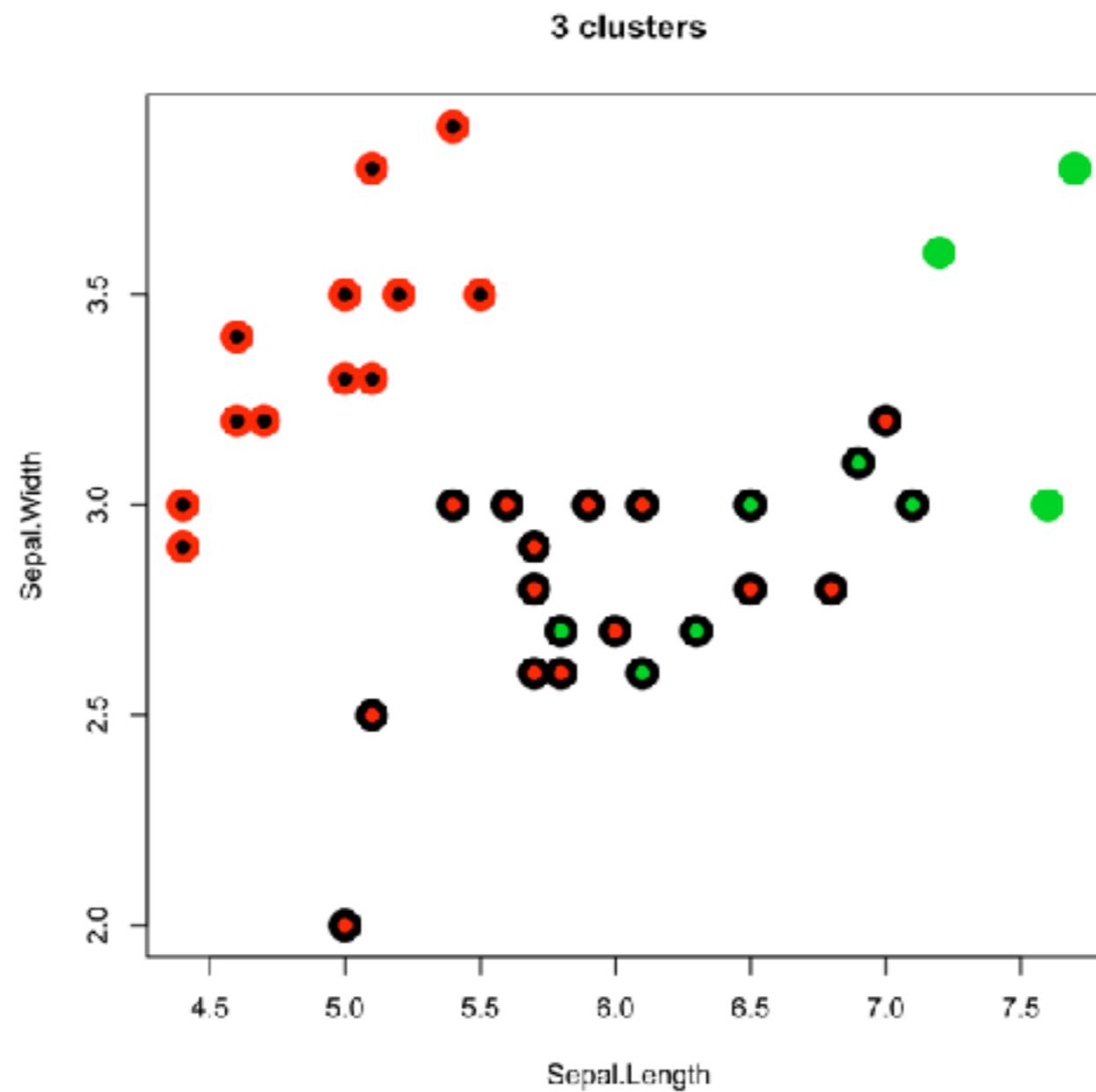
group.3 <- cutree(cluster, k = 3) # prune the tree by 3 clusters

table(group.3, sampleiris$Species) # compare with known classes
```

| group.3 | setosa | versicolor | virginica |
|---------|--------|------------|-----------|
| 1       | 0      | 15         | 9         |
| 2       | 13     | 0          | 0         |
| 3       | 0      | 0          | 3         |

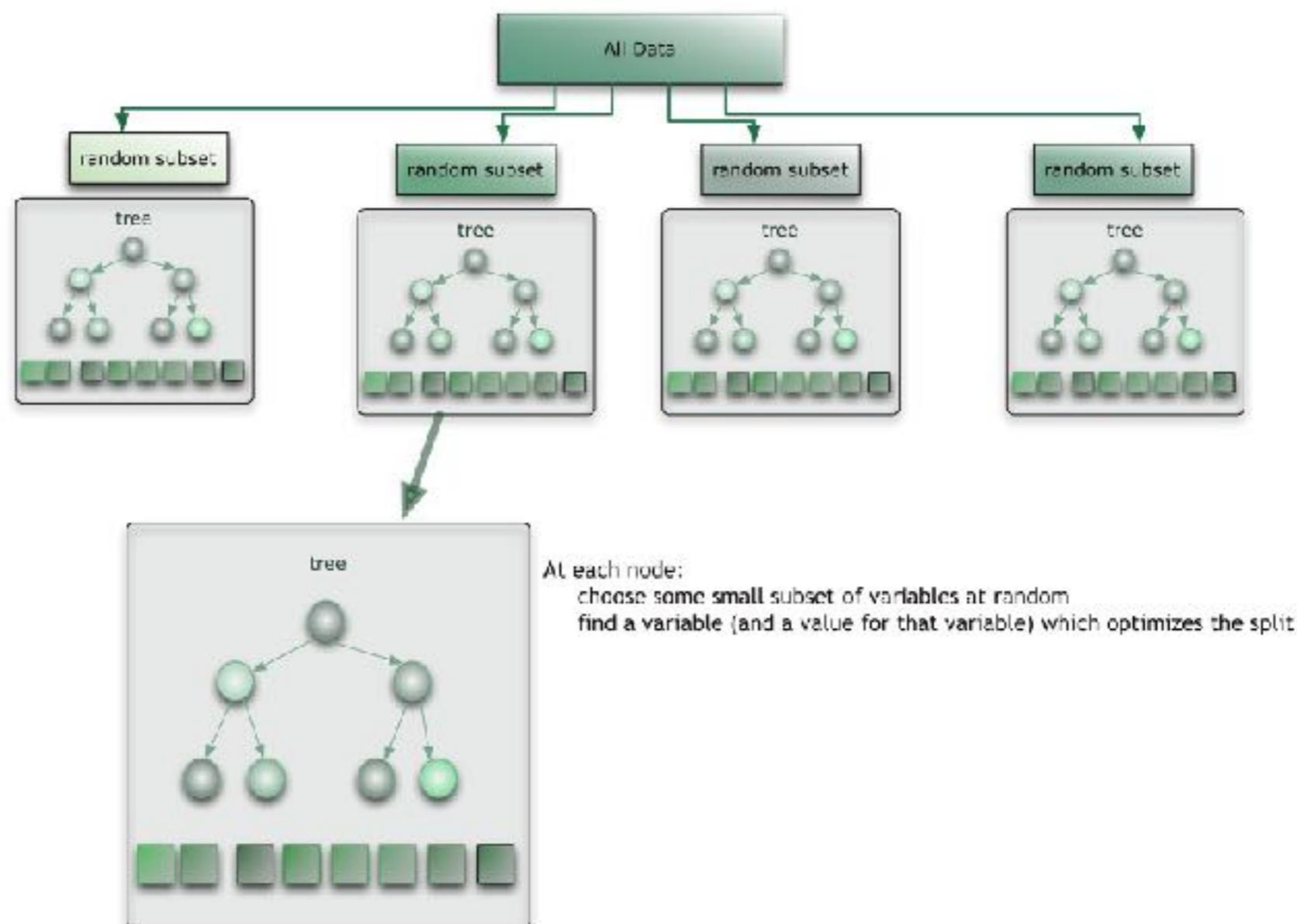
> |

```
plot(sampleiris[,c(1,2)], col=group.3, pch=19, cex=2.5, main="3 clusters")
points(sampleiris[,c(1,2)], col=sampleiris$Species, pch=19, cex=1)
```



# Ensemble : Bagging

## Random Forest



# Random Forest

Here is how such a system is trained; for some number of trees  $T$ :

- 1) Sample  $N$  cases at random with replacement to create a subset of the data. The subset should be about 66% of the total set.
- 2) At each node:
  - a) For some number  $m$  (see below),  $m$  predictor variables are selected at random from all the predictor variables.
  - b) The predictor variable that provides the best split, according to some objective function, is used to do a binary split on that node.
  - c) At the next node, choose another  $m$  variables at random from all predictor variables and do the same.

# Bagging

```
> library(randomForest)
#Train 100 trees, random selected attributes
> model <- randomForest(Species~, data=iristrain, nTree=500)
#Predict using the forest
> prediction <- predict(model, newdata=iristest, type='class')
> table(prediction, iristest$Species)
> importance(model)
 MeanDecreaseGini
Sepal.Length 7.807602
Sepal.Width 1.677239
Petal.Length 31.145822
Petal.Width 38.617223
```

## Confusion Matrix and Statistics

| Prediction      | Reference   |                 |                |
|-----------------|-------------|-----------------|----------------|
|                 | Iris-setosa | Iris-versicolor | Iris-virginica |
| Iris-setosa     | 15          | 0               | 0              |
| Iris-versicolor | 0           | 13              | 2              |
| Iris-virginica  | 0           | 0               | 13             |

## Overall Statistics

Accuracy : 0.9535

95% CI : (0.8419, 0.9943)

No Information Rate : 0.3488

P-Value [Acc &gt; NIR] : &lt; 2.2e-16

Kappa : 0.9303

McNemar's Test P-Value : NA

## Statistics by Class:

|                      | Class: Iris-setosa    | Class: Iris-versicolor |
|----------------------|-----------------------|------------------------|
| Sensitivity          | 1.0000                | 1.0000                 |
| Specificity          | 1.0000                | 0.9333                 |
| Pos Pred Value       | 1.0000                | 0.8667                 |
| Neg Pred Value       | 1.0000                | 1.0000                 |
| Prevalence           | 0.3488                | 0.3023                 |
| Detection Rate       | 0.3488                | 0.3023                 |
| Detection Prevalence | 0.3488                | 0.3488                 |
| Balanced Accuracy    | 1.0000                | 0.9667                 |
|                      | Class: Iris-virginica |                        |
| Sensitivity          | 0.8667                |                        |
| Specificity          | 1.0000                |                        |
| Pos Pred Value       | 1.0000                |                        |
| Neg Pred Value       | 0.9333                |                        |
| Prevalence           | 0.3488                |                        |
| Detection Rate       | 0.3023                |                        |
| Detection Prevalence | 0.3023                |                        |
| Balanced Accuracy    | 0.9333                |                        |

# Boosting

```
> library(adabag)
> iris.adaboost <- boosting(Species~, data=trainData, boost=TRUE,
mfinal=5)
> iris.adaboost$class
> table(iris.adaboost$class, trainData$Species)
> prediction <- predict(iris.adaboost, newdata=testData)
> table(prediction$class, testData$Species)
> confusionMatrix(prediction$class, testData$Species)
```

## Confusion Matrix and Statistics

| Prediction      | Reference   |                 |                |
|-----------------|-------------|-----------------|----------------|
|                 | Iris-setosa | Iris-versicolor | Iris-virginica |
| Iris-setosa     | 15          | 0               | 0              |
| Iris-versicolor | 0           | 13              | 1              |
| Iris-virginica  | 0           | 0               | 14             |

## Overall Statistics

Accuracy : 0.9767

95% CI : (0.8771, 0.9994)

No Information Rate : 0.3488

P-Value [Acc &gt; NIR] : &lt; 2.2e-16

Kappa : 0.9651

McNemar's Test P-Value : NA

## Statistics by Class:

## Class: Iris-setosa Class: Iris-versicolor

|                      |        |        |
|----------------------|--------|--------|
| Sensitivity          | 1.0000 | 1.0000 |
| Specificity          | 1.0000 | 0.9667 |
| Pos Pred Value       | 1.0000 | 0.9286 |
| Neg Pred Value       | 1.0000 | 1.0000 |
| Prevalence           | 0.3488 | 0.3023 |
| Detection Rate       | 0.3488 | 0.3023 |
| Detection Prevalence | 0.3488 | 0.3256 |
| Balanced Accuracy    | 1.0000 | 0.9833 |

## Class: Iris-virginica

|                      |        |
|----------------------|--------|
| Sensitivity          | 0.9333 |
| Specificity          | 1.0000 |
| Pos Pred Value       | 1.0000 |
| Neg Pred Value       | 0.9655 |
| Prevalence           | 0.3488 |
| Detection Rate       | 0.3256 |
| Detection Prevalence | 0.3256 |
| Balanced Accuracy    | 0.9667 |

# Association Rules (Market Basket Analysis)

**Support:** The fraction of which our item set occurs in our dataset.

**Confidence:** probability that a rule is correct for a new transaction with items on the left.

**Lift:** The ratio by which by the confidence of a rule exceeds the expected confidence.

**Note:** if the lift is 1 it indicates that the items on the left and right are independent



# Support, Confidence and Lift

There are several measures used to understand various aspects of associated products.

Let's understand the measures with the help of an example.

- In a store, there are 1000 transactions overall.
- Item A appears in 80 transactions and
- Item B occurs in 100 transactions.
- Items A and B appear in 20 transactions together.



**Support** is the ratio of number of times two or more items occur together to the total number of transactions.

- **Support of A** =  $\Pr(A) = 80/1000 = 8\%$  and
- **Support of B** =  $\Pr(B) = 100/1000 = 10\%$ .

**Confidence** is a conditional probability that a randomly selected transaction will include Item A given Item B.

- **Confidence of A** =  $\Pr(A/B) = 20/100 = 20\%$ .

**Lift** can be expressed as the ratio of the probability of Items A and B occurring together to the multiple of the two individual probabilities for Item A and Item B.

- **Lift** =  $\Pr(A,B) / \Pr(A).\Pr(B) = (20/1000)/((80/1000)\times(100/1000)) = 2.5$ .

# How would you use Support, Confidence and Lift?

**Support** of a product or product bundle indicates the popularity of the product or product bundle in the transaction set. Higher the support, more popular is the product or product bundle. This measure can help in identifying driver of traffic to the store. Hence, if Barbie dolls have a higher support then they can be attractively priced to attract traffic to a store.

**Confidence** can be used for product placement strategy and increasing profitability. Place high-margin items with associated high selling (driver) items. If Market Basket Analysis indicates that customers who bought high selling Barbie dolls also bought high-margin candies, then candies should be placed near Barbie dolls.

**Lift** indicates the strength of an association rule over the random co-occurrence of Item A and Item B, given their individual support. Lift provides information about the change in probability of Item A in presence of Item B. Lift values greater than 1.0 indicate that transactions containing Item B tend to contain Item A more often than transactions that do not contain Item B.

# Apriori Algorithm

?apriori

## Usage

```
apriori(data, parameter = NULL, appearance = NULL, control = NULL)
```

## Arguments

**data**

object of class [transactions](#) or any data structure which can be coerced into [transactions](#) (e.g., a binary matrix or data.frame).

**parameter**

object of class [APparameter](#) or named list. The default behavior is to mine rules with support 0.1, confidence 0.8, and maxlen 10.

**appearance**

object of class [APappearance](#) or named list. With this argument item appearance can be restricted. By default all items can appear unrestricted.

**control**

object of class [APcontrol](#) or named list. Controls the performance of the mining algorithm (item sorting, etc.)

# Apriori Algorithm

So lets get started by loading up our libraries and data set.

```
Load the libraries
```

```
library(arules)
```

```
library(arulesViz)
```

```
library(datasets)
```

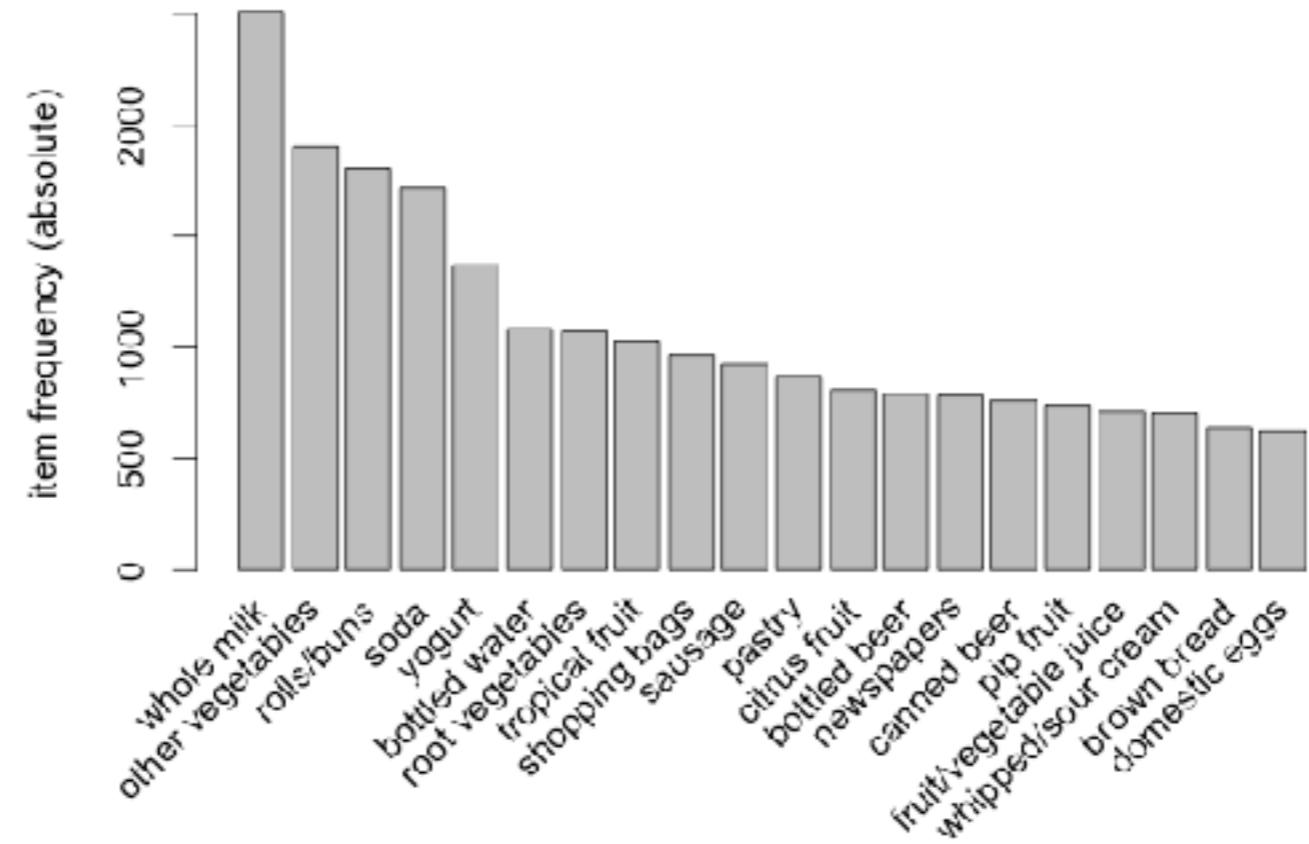
```
Load the data set
```

```
data(Groceries)
```

# Explore Data

# Create an item frequency plot for the top 20 items

```
itemFrequencyPlot(Groceries, topN=20,type="absolute")
```



```
rules <- apriori(Groceries, parameter = list(supp = 0.001, conf = 0.8))

Show the top 5 rules, but only 2 digits

options(digits=2)

inspect(rules[1:5])
```

| lhs                                             | rhs    | support | confidence | lift |
|-------------------------------------------------|--------|---------|------------|------|
| 1 {liquor,<br>red/blush wine} => {bottled beer} | 0.0019 | 0.90    | 11.2       |      |
| 2 {curd,<br>cereals} => {whole milk}            | 0.0010 | 0.91    | 3.6        |      |
| 3 {yogurt,<br>cereals} => {whole milk}          | 0.0017 | 0.81    | 3.2        |      |
| 4 {butter,<br>jam} => {whole milk}              | 0.0010 | 0.83    | 3.3        |      |
| 5 { soups,<br>bottled beer} => {whole milk}     | 0.0011 | 0.92    | 3.6        |      |
| >                                               |        |         |            |      |

```
summary(rules)
```

set of 410 rules

rule length distribution (lhs + rhs):sizes

| 3  | 4   | 5   | 6  |
|----|-----|-----|----|
| 29 | 229 | 140 | 12 |

| Min. | 1st Qu. | Median | Mean | 3rd Qu. | Max. |
|------|---------|--------|------|---------|------|
| 3.0  | 4.0     | 4.0    | 4.3  | 5.0     | 6.0  |

summary of quality measures:

| support         | confidence   | lift         |
|-----------------|--------------|--------------|
| Min. :0.00102   | Min. :0.80   | Min. : 3.1   |
| 1st Qu.:0.00102 | 1st Qu.:0.83 | 1st Qu.: 3.3 |
| Median :0.00122 | Median :0.85 | Median : 3.6 |
| Mean   :0.00125 | Mean   :0.87 | Mean   : 4.0 |
| 3rd Qu.:0.00132 | 3rd Qu.:0.91 | 3rd Qu.: 4.3 |
| Max.   :0.00315 | Max.   :1.00 | Max.   :11.2 |

mining info:

| data      | ntransactions | support | confidence |
|-----------|---------------|---------|------------|
| Groceries | 9835          | 0.001   | 0.8        |

## # Sort Rules

```
rules<-sort(rules, by="confidence", decreasing=TRUE)
```

| lhs                                                  | rhs             | support | confidence | lift |
|------------------------------------------------------|-----------------|---------|------------|------|
| 1 {rice,<br>sugar}                                   | => {whole milk} | 0.0012  | 1          | 3.9  |
| 2 {canned fish,<br>hygiene articles}                 | => {whole milk} | 0.0011  | 1          | 3.9  |
| 3 {root vegetables,<br>butter,<br>rice}              | => {whole milk} | 0.0010  | 1          | 3.9  |
| 4 {root vegetables,<br>whipped/sour cream,<br>flour} | => {whole milk} | 0.0017  | 1          | 3.9  |
| 5 {butter,<br>soft cheese,<br>domestic eggs}         | => {whole milk} | 0.0010  | 1          | 3.9  |

# Change to have limit association in

```
change to have maximum of 3
rules <- apriori(Groceries, parameter = list(supp = 0.001, conf =
0.8,maxlen=3))
inspect(rules[1:5])
```

| lhs                                             | rhs             | support | confidence | lift |
|-------------------------------------------------|-----------------|---------|------------|------|
| 1 {liquor,<br>red/blush wine} => {bottled beer} |                 | 0.0019  | 0.90       | 11.2 |
| 2 {curd,<br>cereals}                            | => {whole milk} | 0.0010  | 0.91       | 3.6  |
| 3 {yogurt,<br>cereals}                          | => {whole milk} | 0.0017  | 0.81       | 3.2  |
| 4 {butter,<br>jam}                              | => {whole milk} | 0.0010  | 0.83       | 3.3  |
| 5 {soups,<br>bottled beer}                      | => {whole milk} | 0.0011  | 0.92       | 3.6  |

## # Rules pruned

```
subset.matrix <- is.subset(rules, rules)
subset.matrix[lower.tri(subset.matrix, diag=T)] <- NA
redundant <- colSums(subset.matrix, na.rm=T) >= 1
rules.pruned <- rules[!redundant]
rules<-rules.pruned
summary(rules)
```

```
set of 330 rules

rule length distribution (lhs + rhs):sizes
 3 4 5 6
 29 216 84 1

 Min. 1st Qu. Median Mean 3rd Qu. Max.
 3.0 4.0 4.0 4.2 5.0 6.0

summary of quality measures:
 support confidence lift
 Min. :0.00102 Min. :0.80 Min. : 3.1
 1st Qu.:0.00102 1st Qu.:0.82 1st Qu.: 3.3
 Median :0.00122 Median :0.85 Median : 3.6
 Mean :0.00127 Mean :0.86 Mean : 3.8
 3rd Qu.:0.00132 3rd Qu.:0.91 3rd Qu.: 4.3
 Max. :0.00315 Max. :1.00 Max. :11.2

mining info:
 data ntransactions support confidence
 Groceries 9835 0.001 0.8
```

# Targeting Items

What are customers likely to buy before buying whole milk?

What are customers likely to buy if they purchase whole milk?

This essentially means we want to set either the Left Hand Side and Right Hand Side. This is not difficult to do with R!

# Find whole milk's antecedents

```
rules<-apriori(data=Groceries, parameter=list(supp=0.001,conf =
0.08), appearance = list(default="lhs",rhs="whole milk"), control =
list(verbose=F))
```

```
rules<-sort(rules, decreasing=TRUE,by="confidence")
```

```
inspect(rules[1:5])
```

|   | lhs                                                | rhs             | support | confidence | lift |
|---|----------------------------------------------------|-----------------|---------|------------|------|
| 1 | {rice,<br>sugar}                                   | => {whole milk} | 0.0012  | 1          | 3.9  |
| 2 | {canned fish,<br>hygiene articles}                 | => {whole milk} | 0.0011  | 1          | 3.9  |
| 3 | {root vegetables,<br>butter,<br>rice}              | => {whole milk} | 0.0010  | 1          | 3.9  |
| 4 | {root vegetables,<br>whipped/sour cream,<br>flour} | => {whole milk} | 0.0017  | 1          | 3.9  |
| 5 | {butter,<br>soft cheese,<br>domestic eggs}         | => {whole milk} | 0.0010  | 1          | 3.9  |
| > |                                                    |                 |         |            |      |

# Likely to buy after buy whole milk

```
rules<-apriori(data=Groceries, parameter=list(supp=0.001,conf =
0.15,minlen=2), appearance = list(default="rhs",lhs="whole milk"), control =
list(verbose=F))

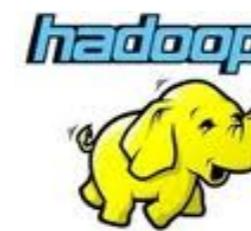
rules<-sort(rules, decreasing=TRUE,by="confidence")

inspect(rules[1:5])
```

|   | lhs             | rhs                | support | confidence | lift |
|---|-----------------|--------------------|---------|------------|------|
| 1 | {whole milk} => | {other vegetables} | 0.075   | 0.29       | 1.5  |
| 2 | {whole milk} => | {rolls/buns}       | 0.057   | 0.22       | 1.2  |
| 3 | {whole milk} => | {yogurt}           | 0.056   | 0.22       | 1.6  |
| 4 | {whole milk} => | {root vegetables}  | 0.049   | 0.19       | 1.8  |
| 5 | {whole milk} => | {tropical fruit}   | 0.042   | 0.17       | 1.6  |
| > |                 |                    |         |            |      |

# Hadoop an Map-reduce Paradigm

MapReduce computing paradigm (E.g., Hadoop) vs. Traditional database systems



vs.

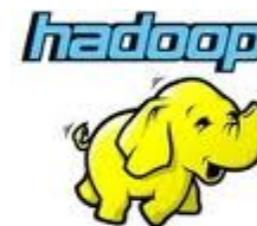


Many enterprises are turning to Hadoop

Especially applications generating big data

Web applications, social networks, scientific applications

# Why Hadoop is able to compete?



vs.



Scalability (petabytes of data, thousands of machines)



Flexibility in accepting all data formats (no schema)



Efficient and simple fault-tolerant mechanism



Commodity inexpensive hardware



Performance (tons of indexing, tuning, data organization tech.)



Features:

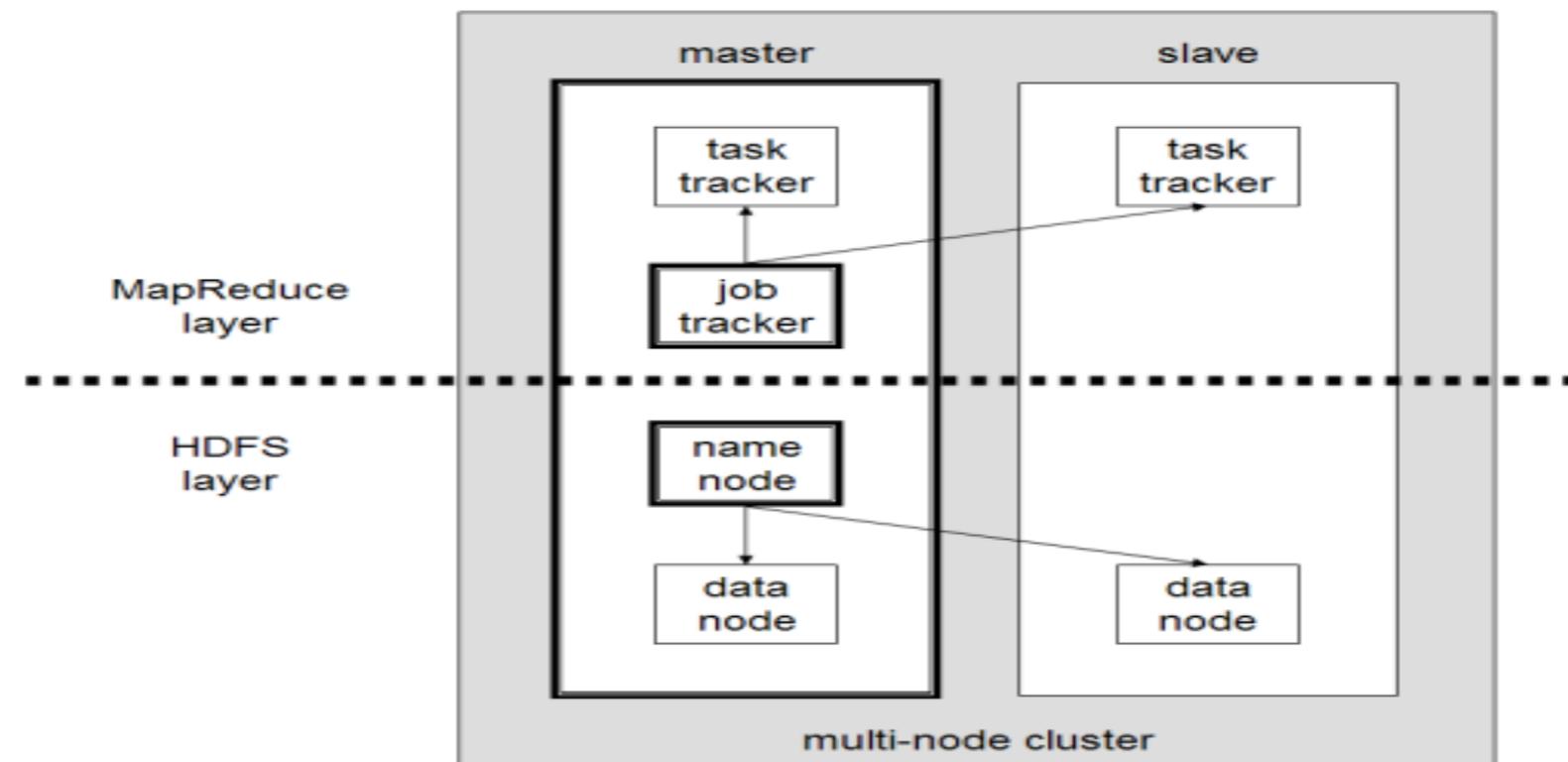
- Provenance tracking
- Annotation management
- ....

# What is Hadoop?

- Hadoop is a software framework for *distributed processing* of *large datasets* across *large clusters* of computers
  - *Large datasets* → Terabytes or petabytes of data
  - *Large clusters* → hundreds or thousands of nodes
- Hadoop is open-source implementation for Google **MapReduce**
- Hadoop is based on a simple programming model called *MapReduce*
- Hadoop is based on a simple data model, *any data will fit*

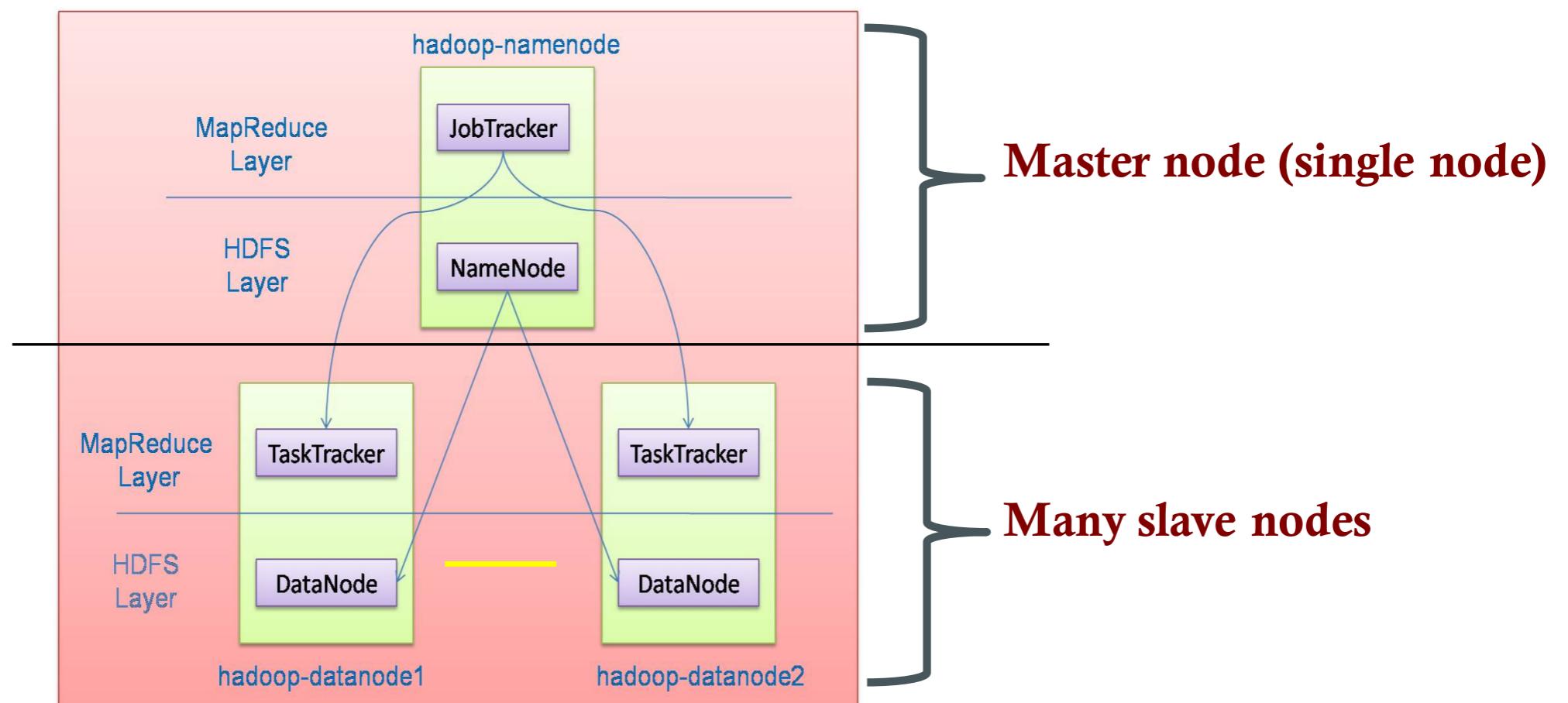
# What is Hadoop?

- **Hadoop framework consists on two main layers**
  - Distributed file system (HDFS)
  - Execution engine (MapReduce)



# Hadoop Architecture

- Hadoop is designed as a *master-slave shared-nothing* architecture



# Design principles of Hadoop

- Need to process big data
- Need to parallelize computation across thousands of nodes
- **Commodity hardware**
  - Large number of low-end cheap machines working in parallel to solve a computing problem
- This is in contrast to **Parallel DBs**
  - Small number of high-end expensive machines

# Design principles of Hadoop

- **Automatic parallelization & distribution**
  - Hidden from the end-user
- **Fault tolerance and automatic recovery**
  - Nodes/tasks will fail and will recover automatically
- **Clean and simple programming abstraction**
  - Users only provide two functions “map” and “reduce”

# RHadoop

```
install.packages(c('rJava','RJSONIO', 'itertools',
'digest','Rcpp','httr','functional','devtools', 'plyr','reshape2')

Sys.setenv("HADOOP_CMD"="/usr/local/Cellar/hadoop/2.7.1/bin/
hadoop")

Sys.setenv("HADOOP_STREAMING"="/usr/local/Cellar/hadoop/
2.7.1/libexec/share/hadoop/tools/lib/hadoop-streaming-2.7.1.jar")

Sys.getenv("HADOOP_CMD")

Sys.setenv("HADOOP_HOME"="/usr/local/Cellar/hadoop/2.7.1")
```

# Install RHadoop

Installing RHadoop [rhdfs, rmr, rhbase]

1. Download RHadoop packages from GitHub repository of Revolution Analytics: <https://github.com/RevolutionAnalytics/RHadoop>

rmr: [rmr-2.2.2.tar.gz]

rhdfs: [rhdfs-1.6.0.tar.gz]

rhbase: [rhbase-1.2.0.tar.gz]

2. Installing packages.

For rmr we use: R CMD INSTALL rmr-2.2.2.tar.gz

For rhdfs we use: R CMD INSTALL rmr-2.2.2.tar.gz

For rhbase we use: R CMD INSTALL rhbase-1.2.0.tar.gz

# gdp data

```
library(rmr2)
library(rhdfs)
gdp <- NA
gdp <- read.csv("~/Downloads/GDP.csv")
gdp <- gdp[,1:4]
gdp$GDP <- as.double(gsub(",","",gdp$GDP))
head(gdp)
```

# Setup Map-Reduce Function

```
hdfs.init()
```

```
gdp.values <- to.dfs(gdp)
```

```
aaplRevenue = 181890
```

```
gdp.map.fn <- function(k,v) {
```

```
 key <- ifelse(v[4] < aaplRevenue, "less", "greater")
```

```
 keyval(key, 1)
```

```
}
```

```
count.reduce.fn <- function(k,v) {
```

```
 keyval(k, length(v))
```

# Run Map-Reduce Function

```
count <- mapreduce(input=gdp.values, map = gdp.map.fn, reduce =
count.reduce.fn)
from.dfs(count)
```

# Data Science in Practice Labs

## Titanic Survival Prediction



# Titanic Case Studies

```
> titanic.train <- read.csv("titanic-train.csv", stringsAsFactors=FALSE)
> titanic.test <- read.csv("titanic-test.csv", stringsAsFactors=FALSE)

> titanic.test$Survived <- rep(0, 418)
```

## Creating Submit File

```
> titanic.submit <- data.frame(PassengerId = titanic.test$PassengerId,
 Survived = titanic-test$Survived)

> write.csv(submit, file = "titanic-submit.csv", row.names = FALSE)
```

# Reproducible Research in R

- Sweave
- KnitR
- KnitR + pandoc
- Rpub
- Shiny

# Typical LATEX

```
\documentclass{article}
\usepackage{times}

\begin{document}

% Article top matter
\title{How to Structure a \LaTeX{} Document}
\author{John Doe}
\date{\today}

Blah blah blah.....

\end{document} %End of document.
```

# Sweave

- R embedded in Latex
- Produce pdf or html files
- R code is run each time, so you are sure the code works
- Document includes results of the code

# Quick Start to Sweave

Insert an R code chunk starting with <<>>=

- Terminate the R code chunk with an @ sign

```
<<easySweave>>=
```

```
x <- mean(1:10)
```

```
print(x)
```

```
@
```

- Save LaTeX with extension ``Rnw''

# Embedding code in text

- To embed a simple R calculation within a document `\Sexpr`

The sum is `\Sexpr{1+2}`

`\Sexpr{paste("result is", 2^x)}`

# Sweave works in a html document

Create a basic html  
document and process  
with Sweave

```
Sweave("filename.rnw",
 driver=RweaveHTML)
```

```
<html>
<head>
<title>Sweave and html</title>
</head>
<body>

Blah blah

<<SweaveCode>>=
1+2
sum(1:10)
@

blah blah
</body>
</html>
```

# KnitR

- knitr ≈ Sweave + cacheSweave + pgfSweave + weaver + R2HTML + more
- The design of knitr allows any input languages (e.g. R, Python and Awk) and any output markup languages (e.g. LaTeX, HTML, Markdown and reStructuredText)
- The name knitr was coined with weave in mind, and it also aims to be neater

# Features of knitR

- Faithful
  - knitr writes everything that you see in an R terminal by default (results, plots and warnings)
- Built-in cache
- Formatting R code.
  - Colors. Uses format R package to “fix code” wrap long lines, add spaces and indent, etc
- Graphics
  - over 20 graphics devices, can set size etc
- Can use custom regular expressions to parse R

# Markdown using knitR

- Markdown is not latex
- Very simple language

eg Emphasis

\*italic\*

\*\*bold\*\*

```
```{r example}
```

```
x <- 1+1rnorm(5)
```

```
```
```

```
```{r}
```

```
plot(1:10)
```

```
hist(rnorm(1000))
```

```
```
```

# Converting MD with Pandoc

- Pandoc a universal document converter
  - <http://johnmacfarlane.net/pandoc/index.html>
  - Easy to convert markdown file to many formats

## **pdf file**

```
system("pandoc -s example.md -t latex -o example.pdf")
```

## **html file**

```
system("pandoc -s example.md -o example.html")
```

## **OpenOffice File**

```
system("pandoc example.md -o example.odt")
```

## **Microsoft Word**

```
system("pandoc example.md -o example.docx")
```

# HTML5 Slides

```
system("pandoc -s -S -i -t dzslides -mathjax slides.md -o slides.html")
```

[http://bcb.dfci.harvard.edu/~aedin/courses/ReproducibleResearch/  
slides.html](http://bcb.dfci.harvard.edu/~aedin/courses/ReproducibleResearch/slides.html)

# Online publishing - Rpubs

- Free, from Rstudio
- Create a new R Markdown Doc
  - File -> New -> R Markdown.
- Click the Knit HTML button
- Preview click Publish

<https://rpubs.com>

# Online Publishing – Shiny

- R package shiny
  - Shiny allows R developers to build simple interactive Web-based interfaces for R scripts, using only R code (no JavaScript required!)

<http://www.rstudio.com/shiny/>

# Introduction to Shiny

- Open Sourced by RStudio November 2012
- Not the first to get R in the browser (rApache, Rserve, Rook)
- Default widgets and settings make it easy to generate apps
- Don't need to know HTML, CSS and javascript to get started
- Twitter Bootstrap for default UI - looks good
- Web sockets for communication between client and server
- Reactive Programming model
- Works on Windows, Mac, Linux

# Introduction to Shiny

Ready to use shiny

- Install R from CRAN
- Useful to have Chrome, Firefox, Safari...
- Rstudio or other texteditor
- Install Shiny using R command: `install.packages("shiny")`

# Simple Example

```
• library(shiny)
• runExample("01_hello")
```



# ui.R :: Controls the look of the App

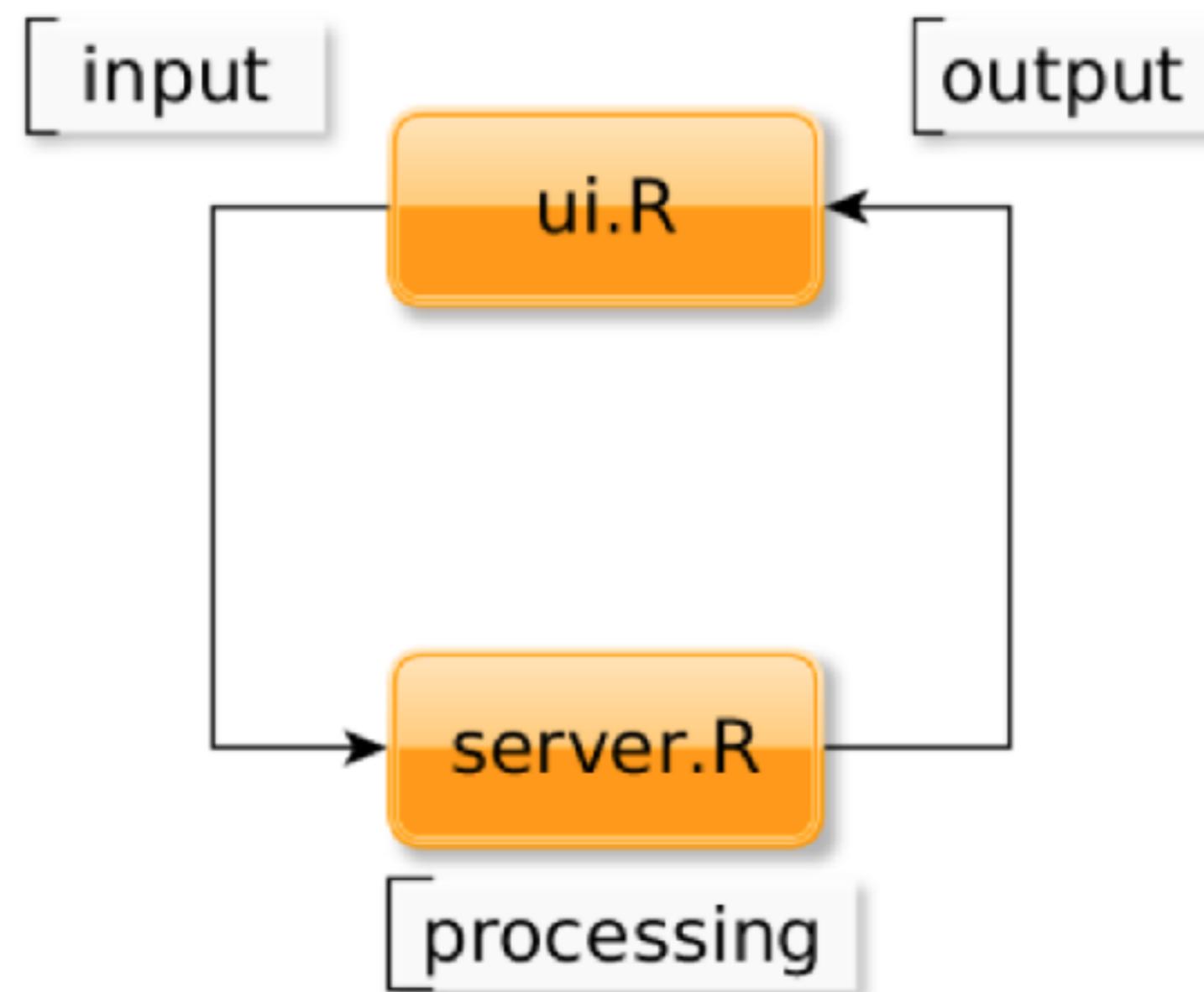
```
library(shiny)

#Define UI for application that plots random distributions
shinyUI(pageWithSidebar(
 #Application title
 headerPanel("HelloShiny!"),
 #Sidebar with a slider input for number of observations
 sidebarPanel(
 sliderInput("obs",
 "Number of observations:",
 min=0, max=1000, value=500
)
),
 #Show a plot of the generated distribution
 mainPanel(plotOutput("distPlot")
)
))
```

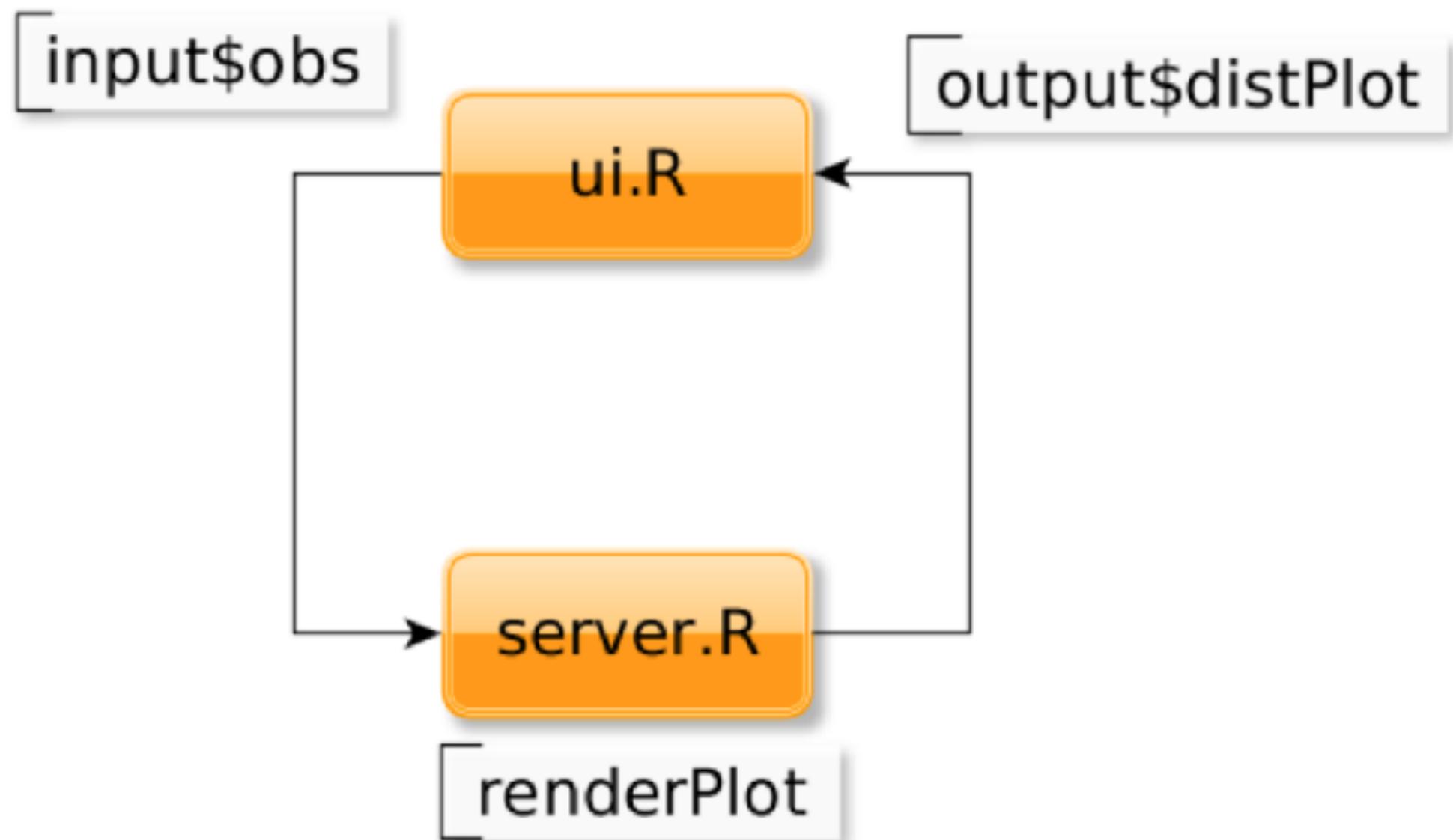
# server.R : Specifies what R is doing

```
#Defineserverlogicrequiredtogenerateandplotarandomdistribution
shinyServer(function(input,output){
 #Expressionthatgeneratesaplotofthedistribution.Theexpression
 #iswrappedinacalltorenderPlottoindicatethat:
 #
 # 1)Itis"reactive"andthereforeshouldbeautomatically reexecutedwheninputschange
 # 2)Itsoutputtypeisaplot
 #
 #
 output$distPlot<-renderPlot({
 #generateanrnormdistributionandplotit
 dist<-rnorm(input$obs)
 hist(dist)
 })
})
```

# Relationship of ui.R and server.R



# Relationship of ui.R and server.R



# Functions for Input

- `sliderInput`: Constructs a slider widget to select a numeric value from a range.
  - `runExample("01_hello")`
- `selectInput`: Create a select list that can be used to choose a single or multiple items from a list of values.
  - `runExample("02_text")`
- `numericInput`: Create an input control for entry of numeric values
  - `runExample("02_text")`
- `checkboxInput`: Create a checkbox that can be used to specify logical values.
  - `runExample("04_mpg")`
- `radioButtons`: Create a set of radio buttons used to select an item from a list.
  - `runExample("06_tabssets")`

# Function for Processing

- **reactive**: Wraps a normal expression to create a reactive expression.  
- `runExample("02_text")`
- **renderPlot**: Renders a reactive plot that is suitable for assigning to an output slot.  
- `runExample("01_hello")`
- **renderPrint**: Makes a reactive version of the given function that captures any printed output.  
- `runExample("02_text")`
- **renderTable**: Creates a reactive table that is suitable for assigning to an output slot  
- `runExample("02_text")`
- **renderText**: Makes a reactive version of the given function that also uses cat to turn its result into a single element character vector.  
- `runExample("03_reactivity")`

# Function for Output

- `plotOutput`: Render a `renderPlot` within an application page.
  - `runExample("01_hello")`
- `tableOutput`: Render a `renderTable` within an application page.
  - `runExample("02_text")`
- `verbatimTextOutput`: Render a reactive output variable as verbatim text within an application page.
  - `runExample("02_text")`
- `textOutput`: Render a reactive output variable as text within an application page.
  - `runExample("02_text")`

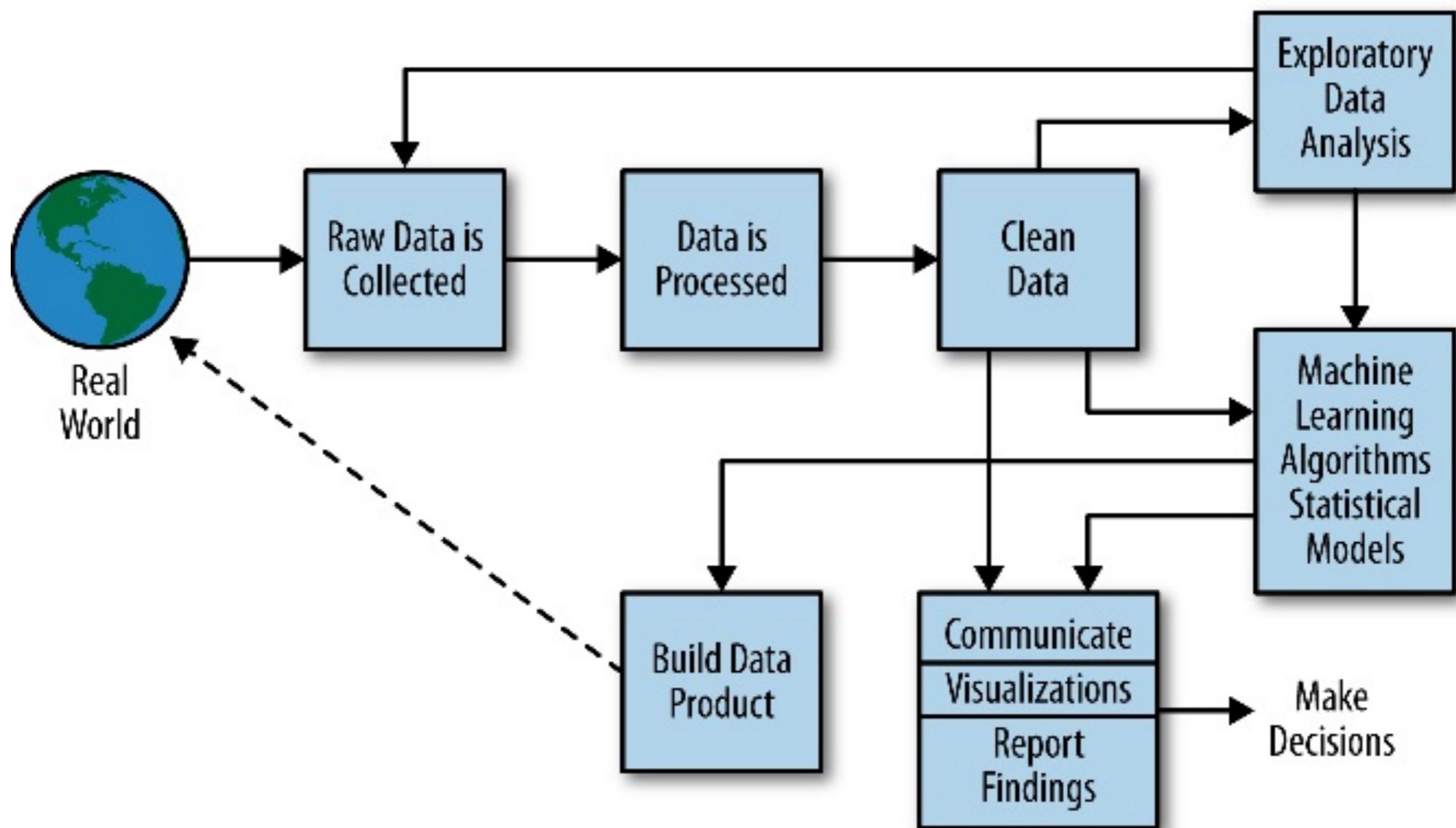
# Advance Features

- Tabs are created by calling the `tabsetPanel` function with a list of tabs created by the `tabPanel` function. Each tab panel is provided a list of output elements which are rendered vertically within the tab.
  - `runExample("06_tabs")`
- UI Enhancements. Shiny offers a variety of functions for including HTML elements directly in pages.
  - `runExample("07_widgets")`
- File upload controls are created by using the `fileInput` function in your ui.R file.
  - access the uploaded data similarly to other types of input: by referring to `input$inputId`.
  - The file contents can be accessed by reading the file named by the `datapath` column
- `runExample("09_upload")`

# Dynamic UI

- The `conditionalPanel` function, which is used in ui.R and wraps a set of UI elements that need to be dynamically shown/hidden
- The `renderUI` function, which is used in server.R in conjunction with the `htmlOutput` function in ui.R, lets you generate calls to UI functions and make the results appear in a predetermined place in the UI
  - `runGist("https://gist.github.com/wch/4034323")`

# Conclusion





# Contact

Contact :

Email: [veerasak.kritsanapraphan@gmail.com](mailto:veerasak.kritsanapraphan@gmail.com)

Twitter: @veerasakk

Blog: <http://dsci.info/blog/>