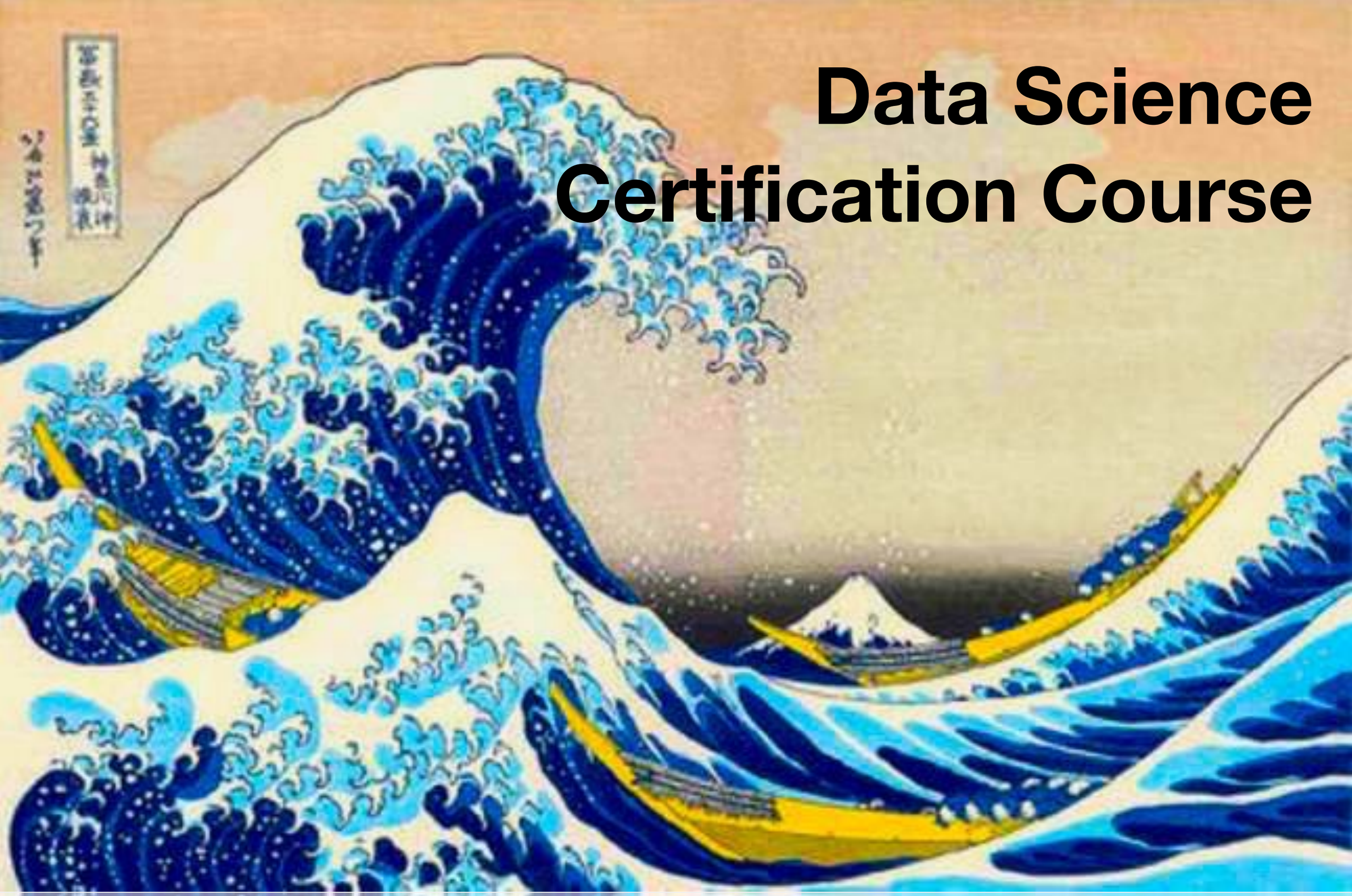


Data Science Certification Course

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)
- Head of Technology Innovation at True Digital Group
- Instructor for IMC and Software Park in Data Science

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?

Module

Module 1: Introduction to Data Science, Big Data and R

Module 2: Data Visualization

Module 3: Data Science: Statistical Analysis

Module 4: Data Science: Machine Learning Algorithm

Module 1 : Introduction to Data Science, Big Data and R

Data Science Certification Course

Kashmir Hill Forbes Staff

Welcome to The Not-So Private Parts where technology & privacy collide

FOLLOW

TECH 2/16/2012 @ 11:02AM | 2,802,865 views

How Target Figured Out A Teen Girl Was Pregnant Before Her Father Did

+ Comment Now + Follow Comments

Share

Next Post

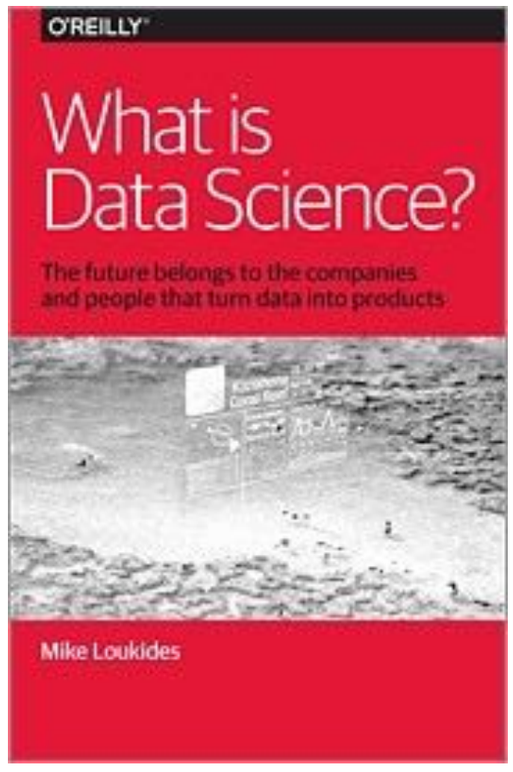
CONFERENCES AND MORE

Every time you go shopping, you share intimate details about your consumption patterns with retailers. And many of those retailers are studying those details to figure out what you like, what you need, and which coupons are most likely to make you happy. Target, for example, has figured out how to data-mine its way into your womb, to figure out whether

Target has got you in its aim

What is Data Science?

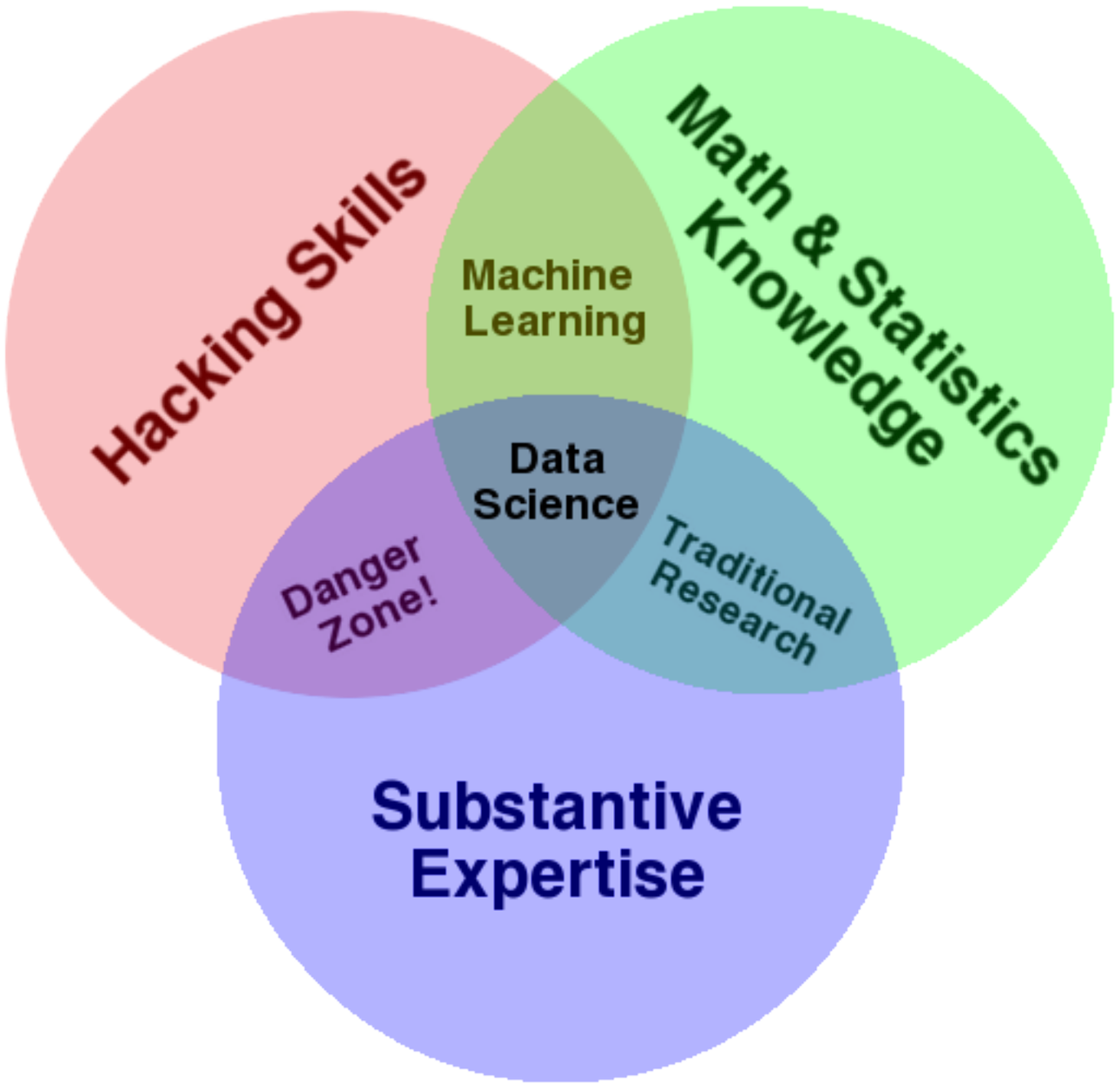
- Data Science aims to derive **knowledge** from **big data**, **efficiently** and **intelligently**
- Data Science encompasses the **set of activities**, **tools**, and **methods** that enable **data-driven activities** in science, business, medicine, and government



http://www.oreillycom/data/free/what-is-data-science.csp "

Goals of Data Science: Turn data into data products.

Data Science - Drew Convey's Definition



http://drewconway.com/zia/2013/3/26/the-data-science-venn-diagram "

Data Science vs. Databases

Element	Databases	Data Science
Data Value	"Precious"	"Cheap"
Data Volume	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

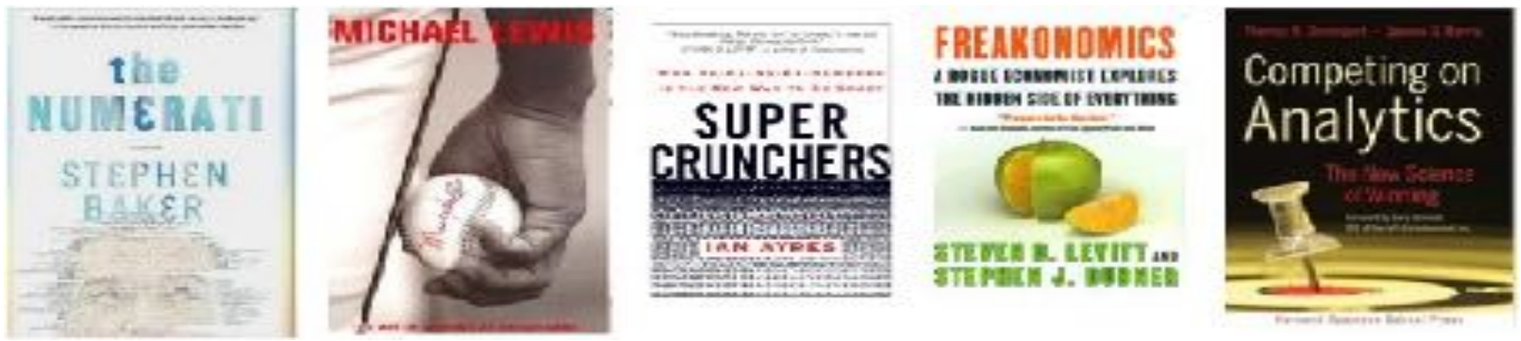
Data Science vs. Databases

Databases	Data Science
Querying the past	Querying the future

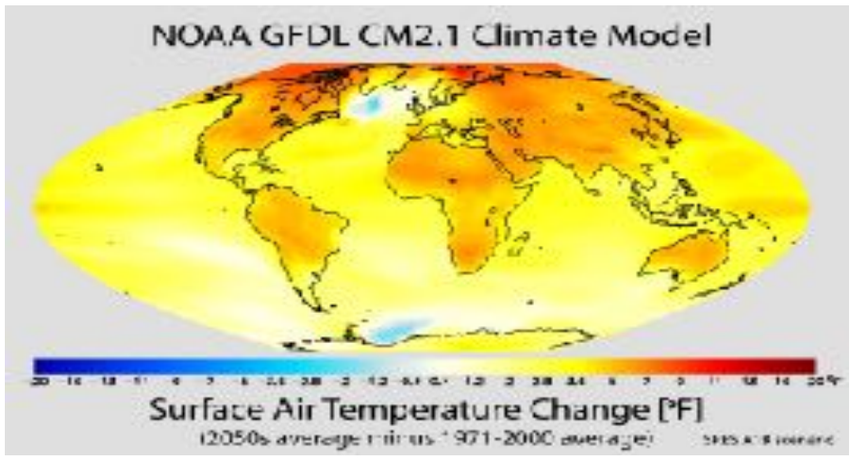
Related – Business Analytics

» 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)

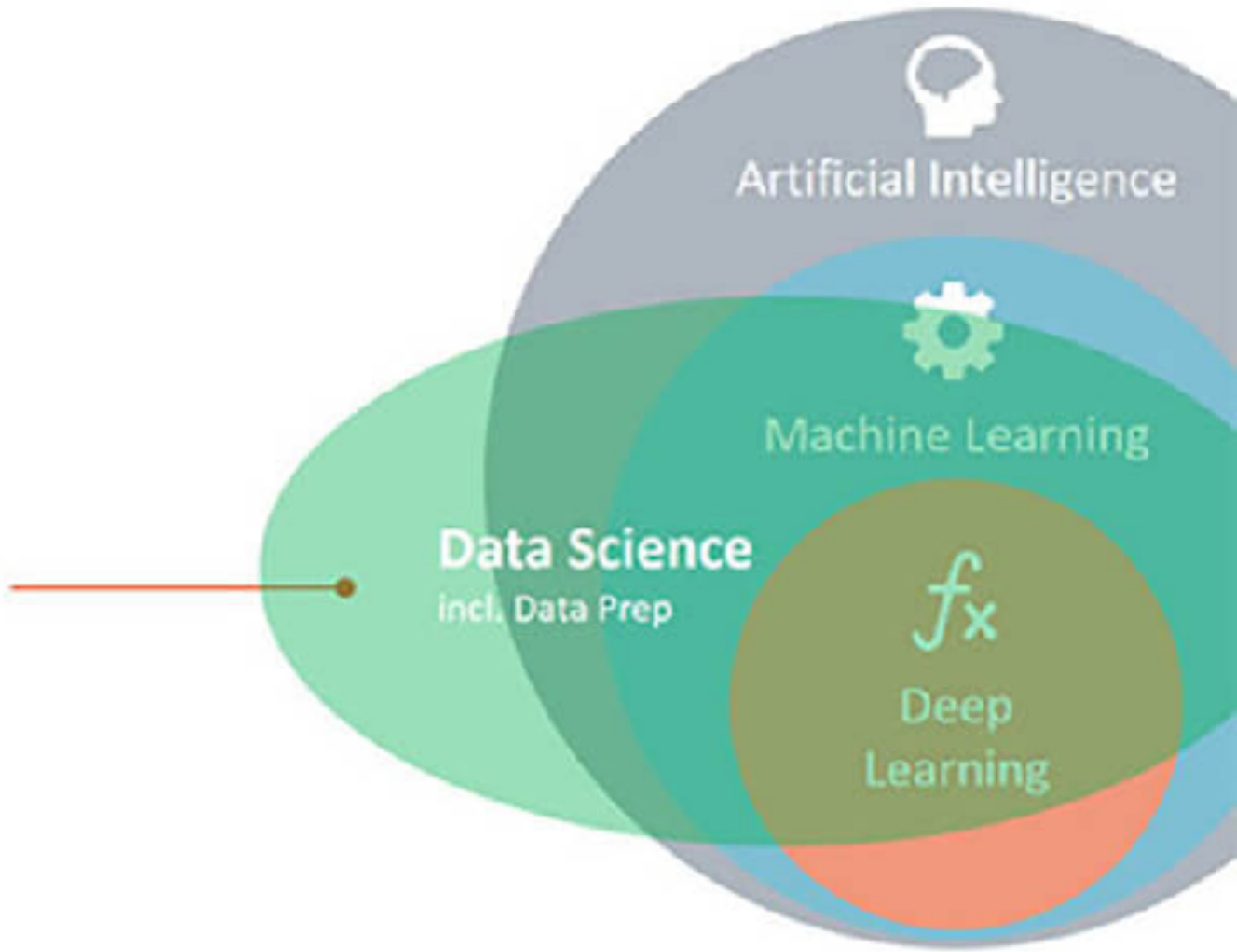
Data Science vs. Traditional Machine

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!

Data Science vs AI vs ML vs DL

Data Science

Covers the practical application of advanced analytics, statistics, machine learning, and the necessary data preparation in a business context.



History of Data Science

- **R. A. Fisher**
  - **» 1935: "The Design of Experiments"**

"correlation does not imply causation"

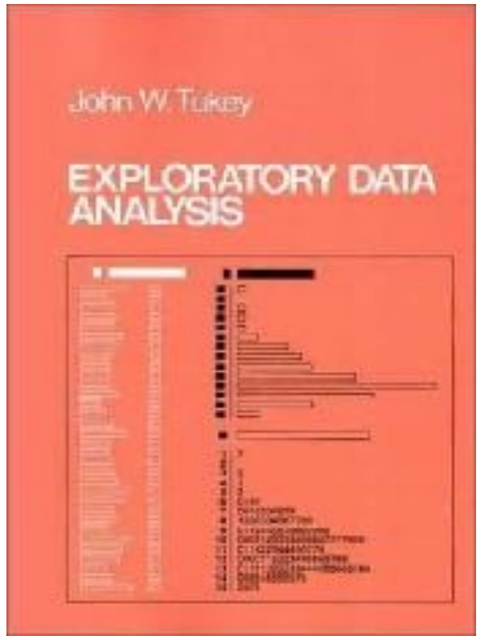
- **W. E. Demming**
  - **» 1939: "Quality Control"**

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

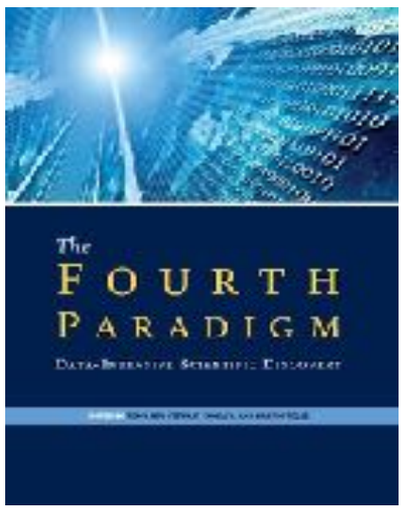
Brief Data Analysis History

- **Tom Mitchell**
  - » **1997: "Machine Learning book"**

- **Google**
  - » **1996: "Prototype Search Engine"**

Google

- **Data-Driven Science eBook**
  - » **2007: "The Fourth Paradigm"**



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"**

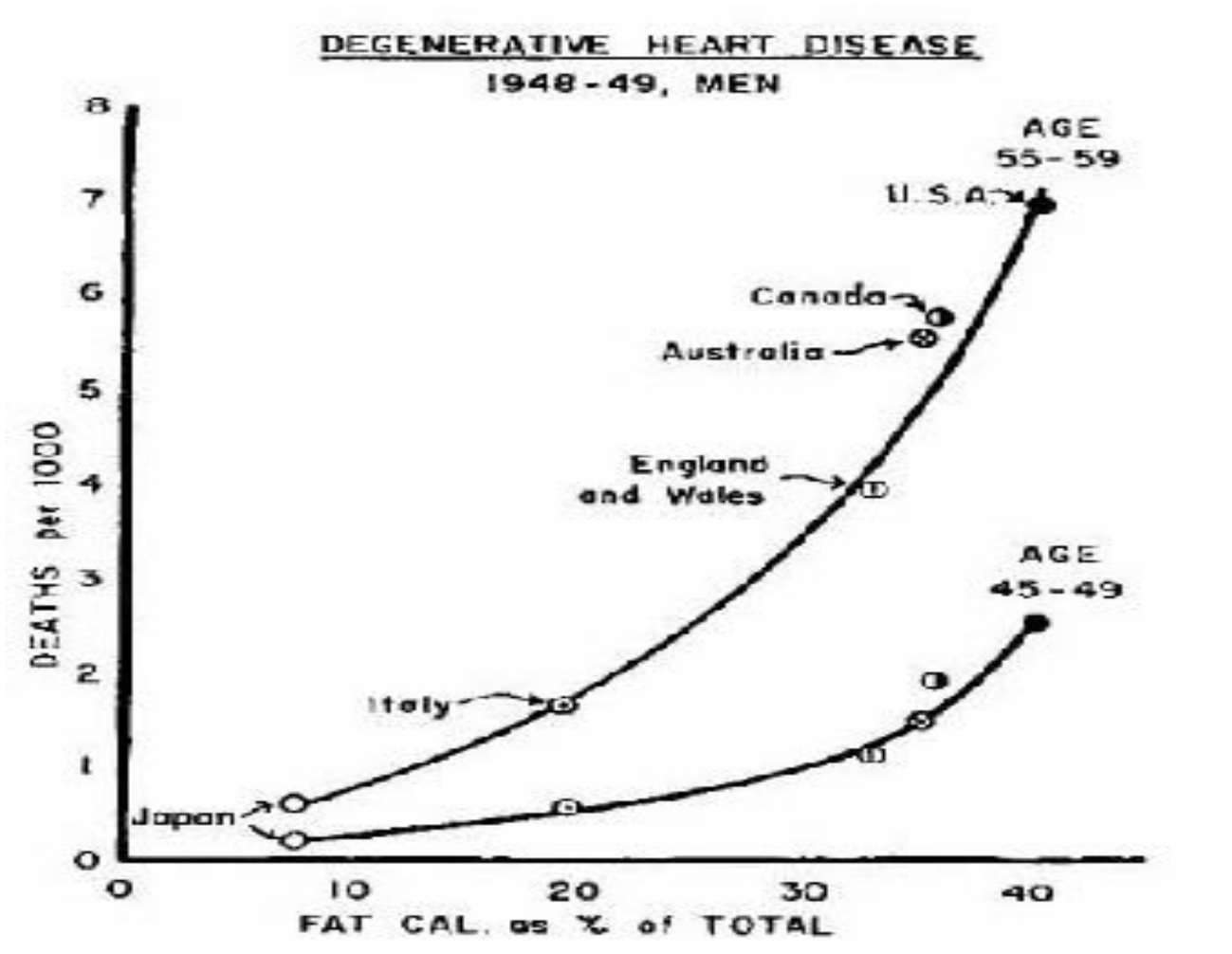
- **Exponential growth in data volume**
  - **» 2010: "The Data Deluge"**

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

Data Science?

- Seven Countries Study (Ancel Keys)

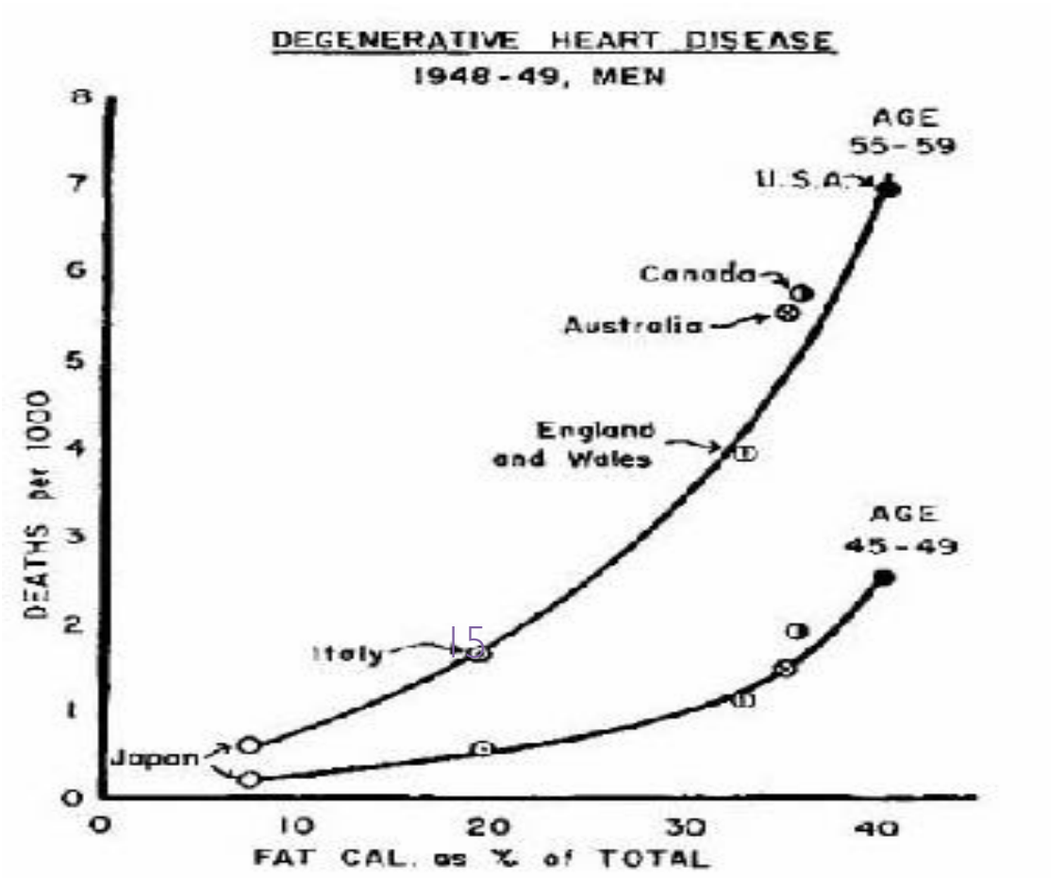
  » Started in 1958, followed13,000 subjects total for 5-40 years



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

Data Science?

- Seven Countries Study (Ancel Keys)
  - » Started in 1958, followed13,000 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?

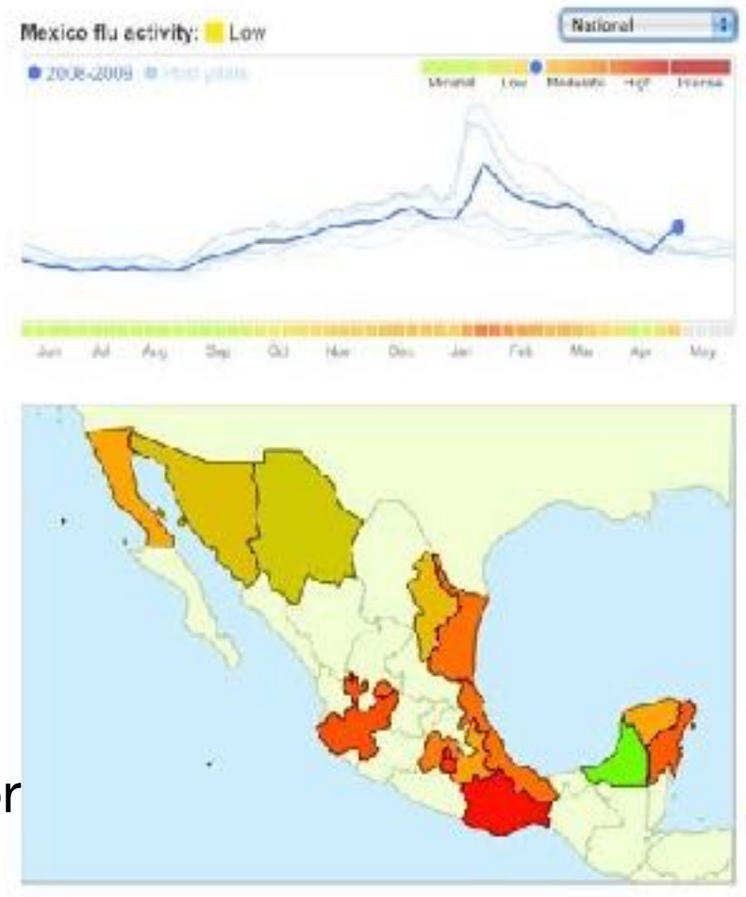
- Nowcasting vs Forecasting

- 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

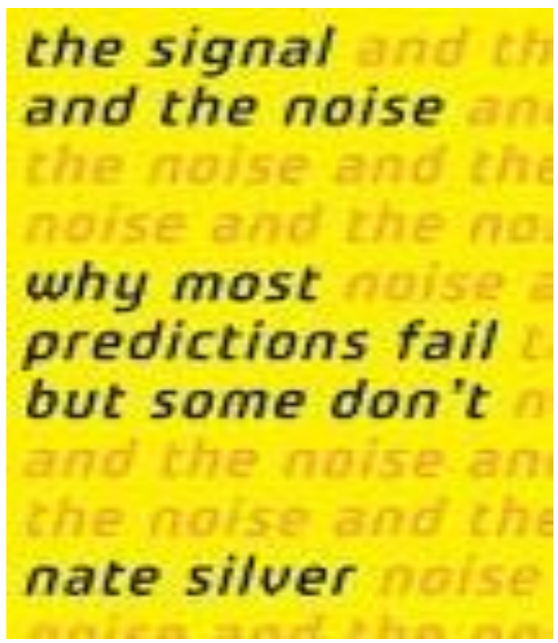
Live results | President | Senate | House | Governor | Choose your

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



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

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

Who will win the presidency?

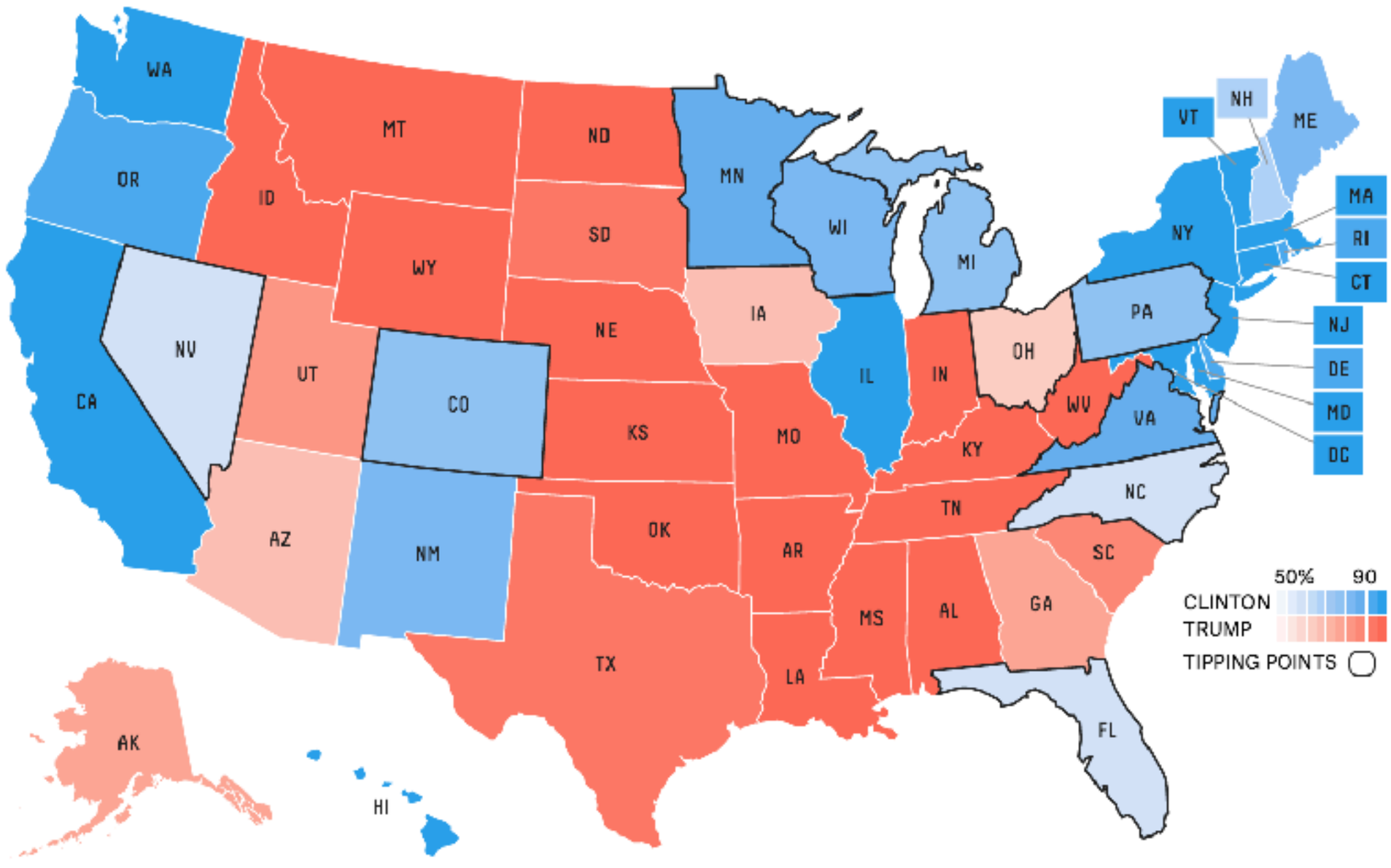
Chance of winning

Hillary Clinton

71.4%

Donald Trump

28.6%



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

BETHANY LECK / GETTY IMAGES

306 trump

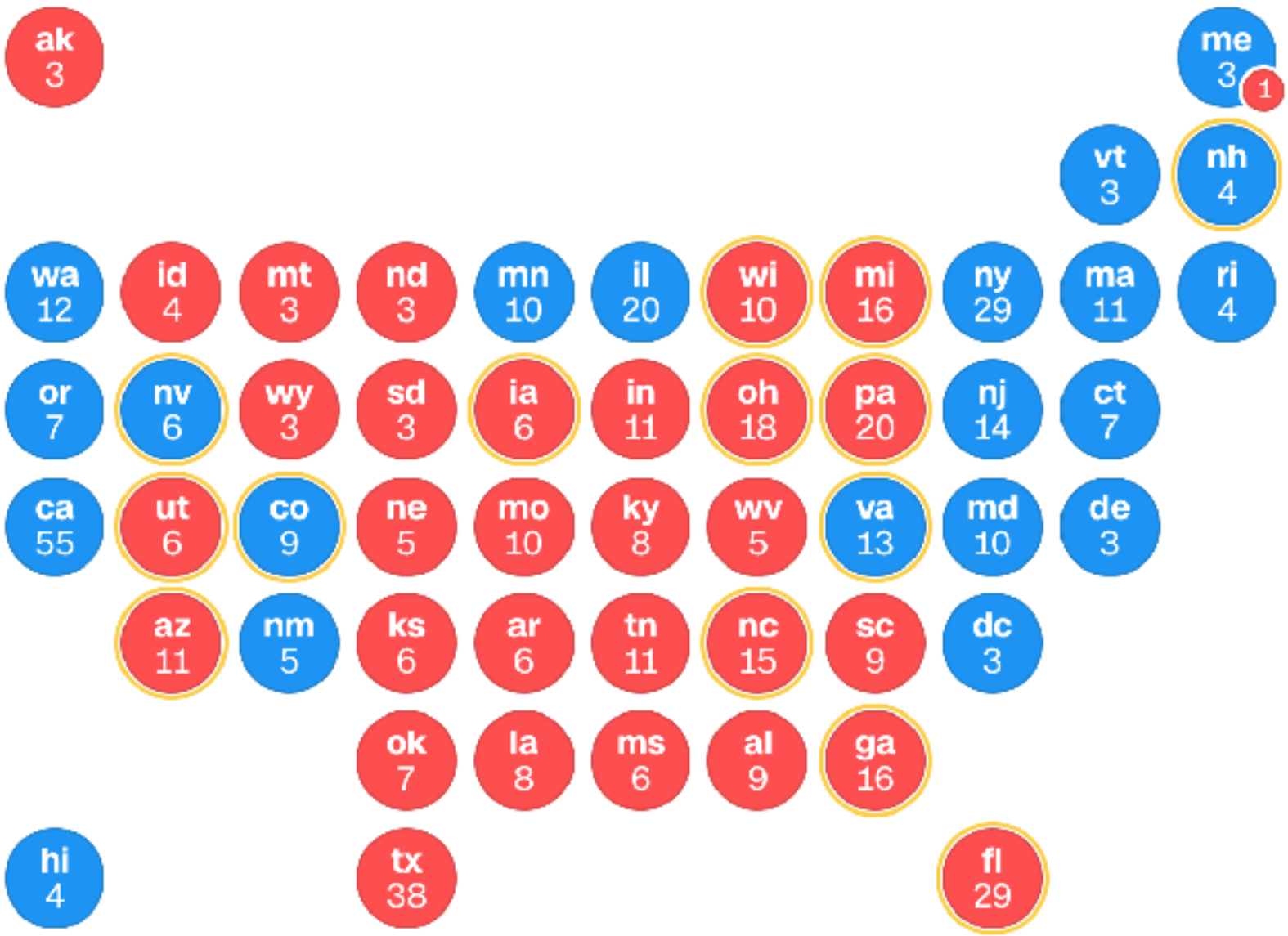
clinton 232

46.2% votes | 62,955,363

270 electoral votes to win

65,788,583 | **48.3% votes**

national map | popular vote



Data Science?

Epidemiological modeling of online social network dynamics

John Cannarella[1], Joshua A. Spechler[1,*]
1 Department of Mechanical and Aerospace Engineering, Princeton University, Princeton, NJ, USA
* E-mail: Corresponding spechler@princeton.edu

Abstract

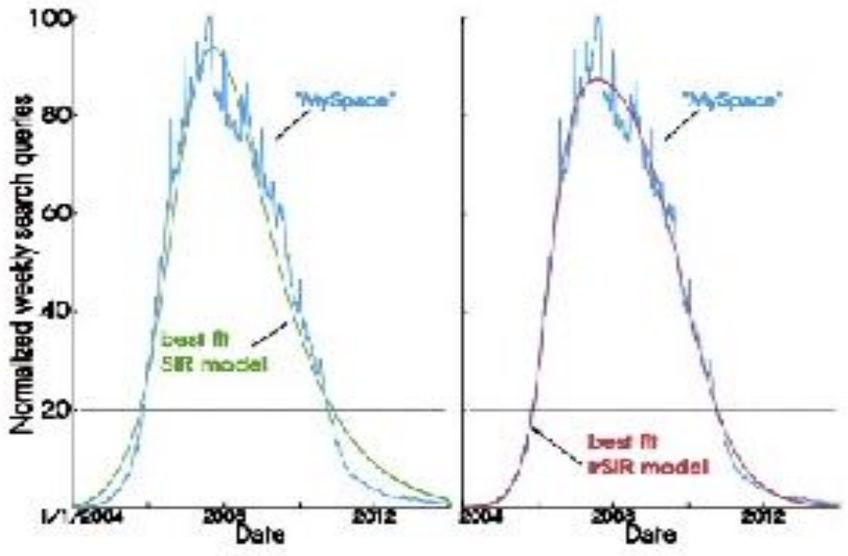
The last decade has seen the rise of immense online social networks (OSNs) such as MySpace and Facebook. In this paper we use epidemiological models to explain user adoption and abandonment of OSNs, where adoption is analogous to infection and abandonment is analogous to recovery. We modify the traditional 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."

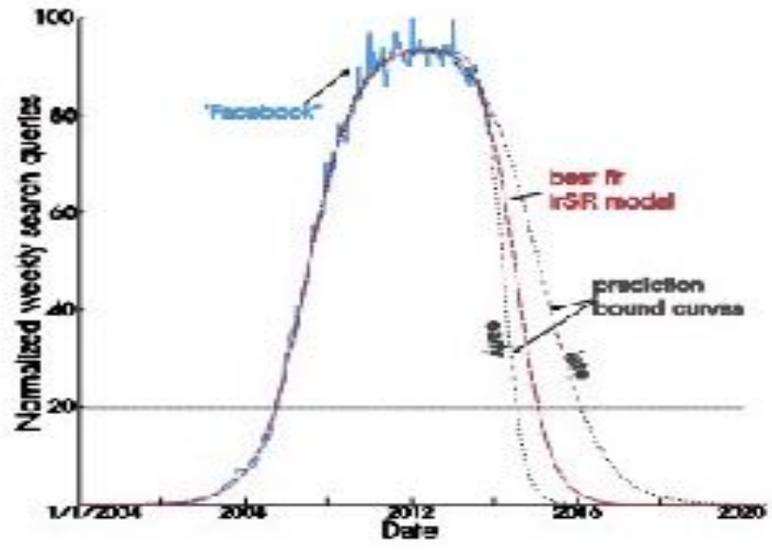
http://arxiv.org/abs/1401.4208

Data Science?

Google Trends searches for
"MySpace"



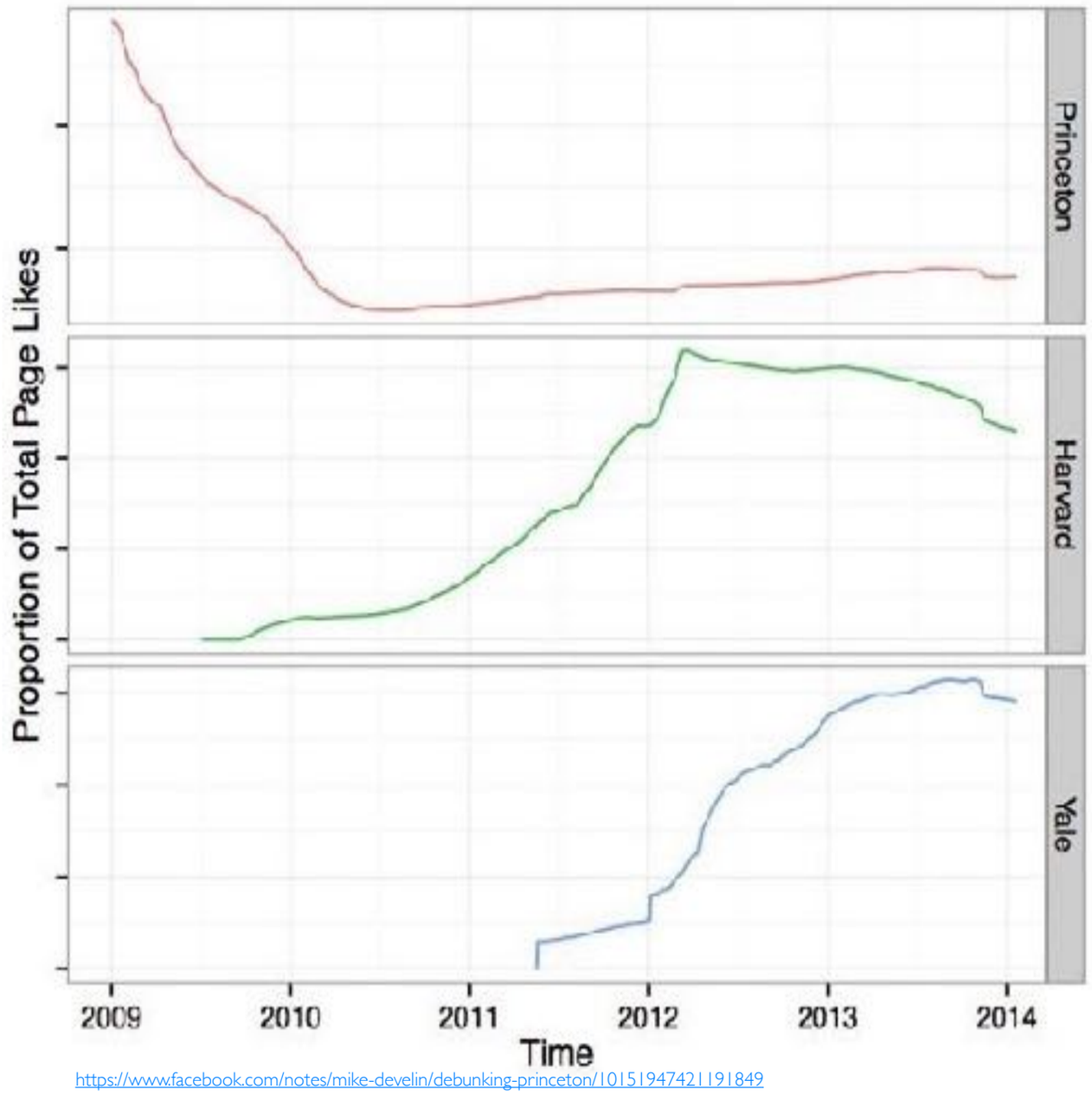
Two Figures from the paper



Searches for
"Facebook"

http://arxiv.org/abs/1401.4208

Data Science?



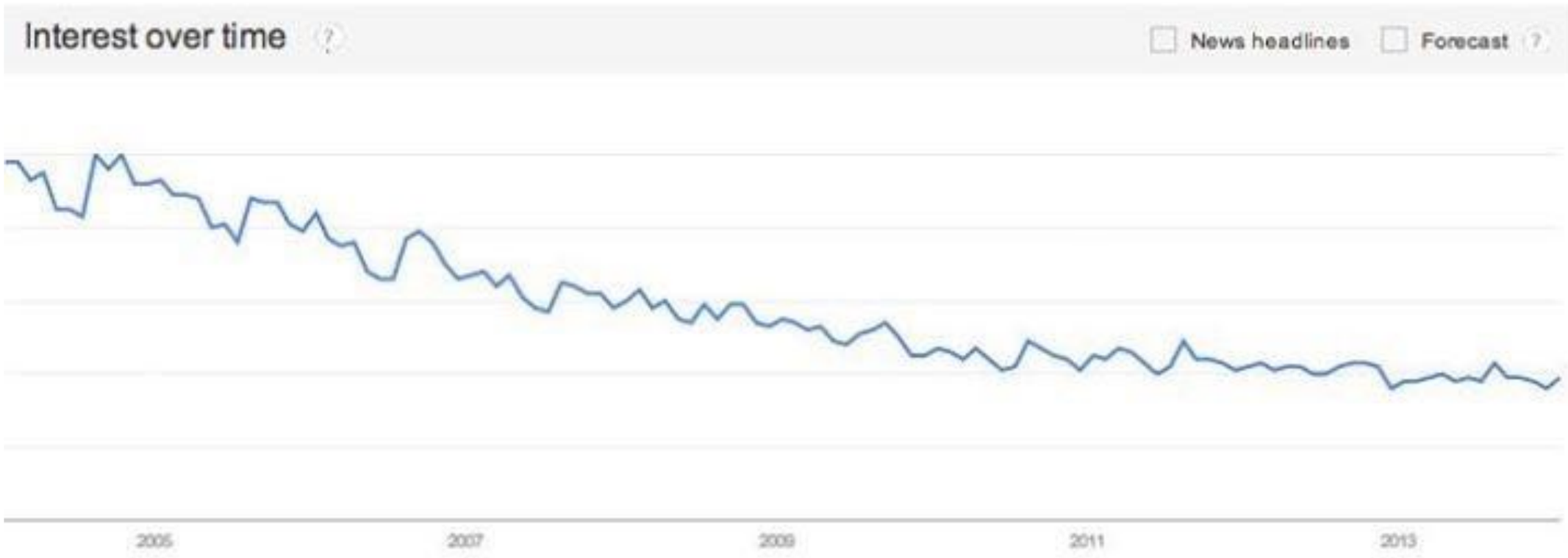
In keeping with the scientific principle "correlation equals causation," our research unequivocally demonstrated that Princeton may be in danger of disappearing entirely.

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

Data Science?

… and based on Princeton search trends:

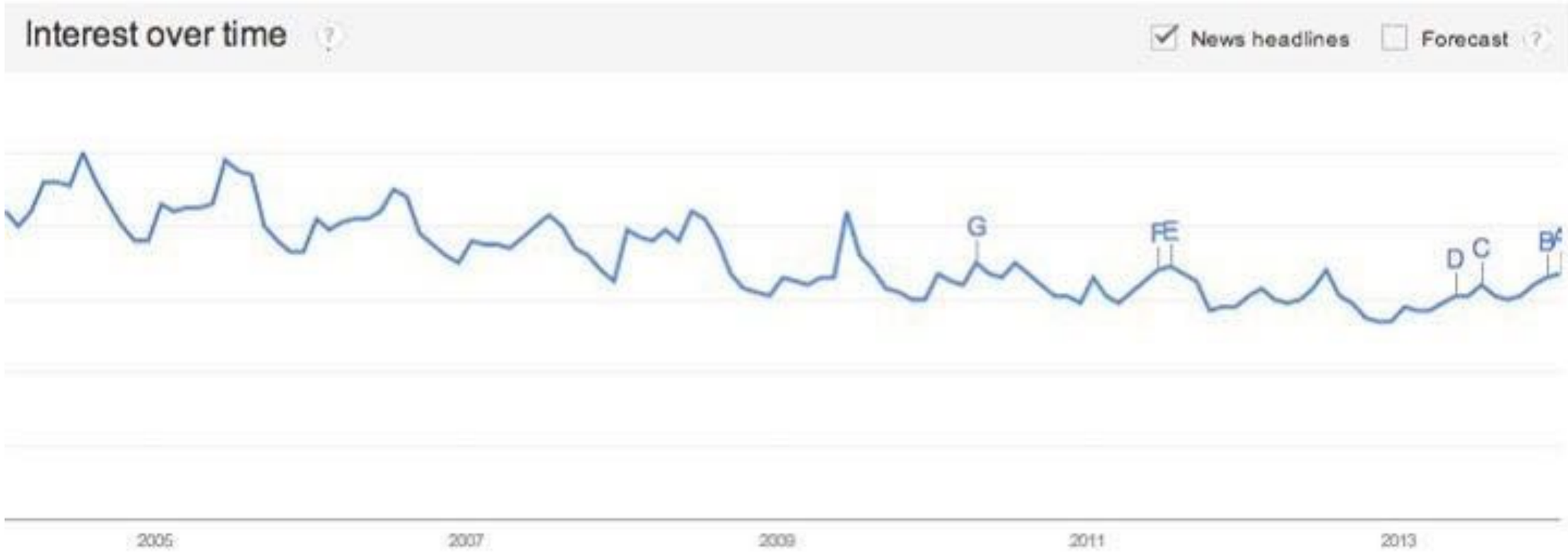
“This trend suggests that Princeton will have only half its current enrollment by 2018, and by 2021 it will have no students at all,…”



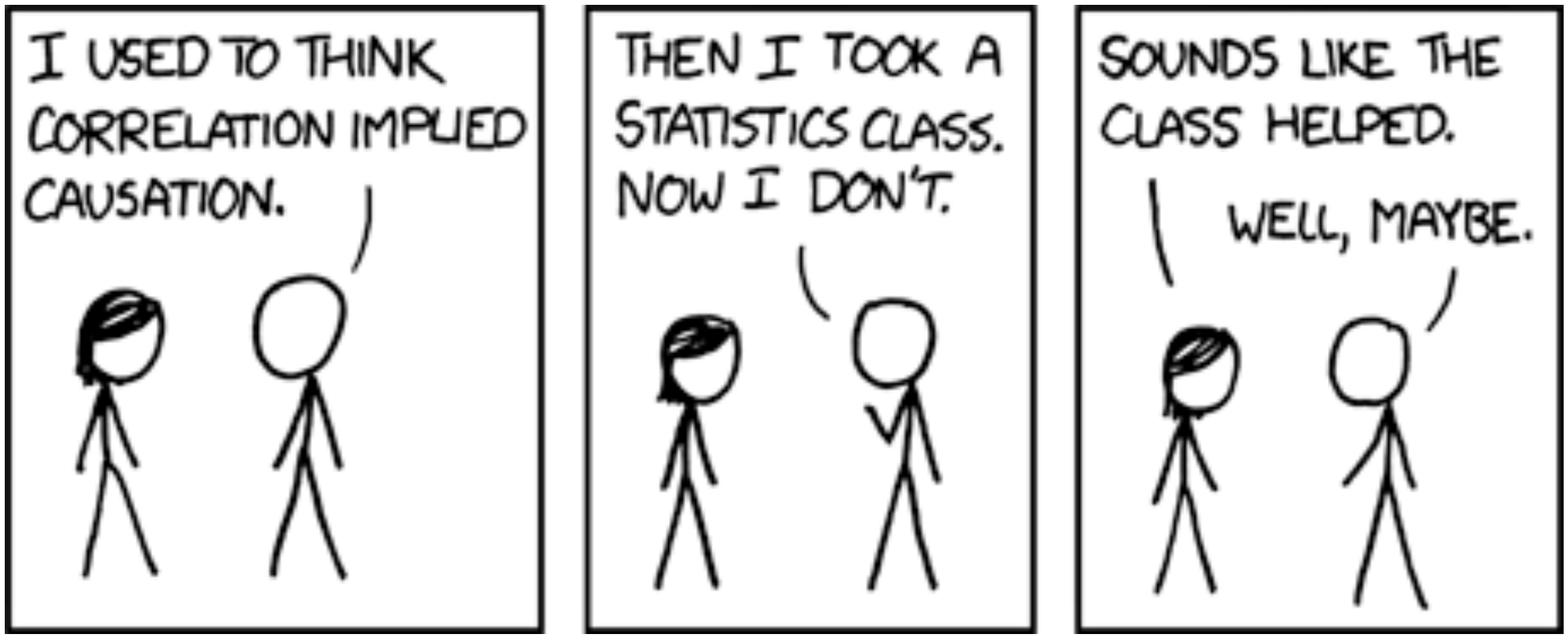
https://www.facebook.com/notes/mike-develin/debunking-princeton/10151947421191849

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:



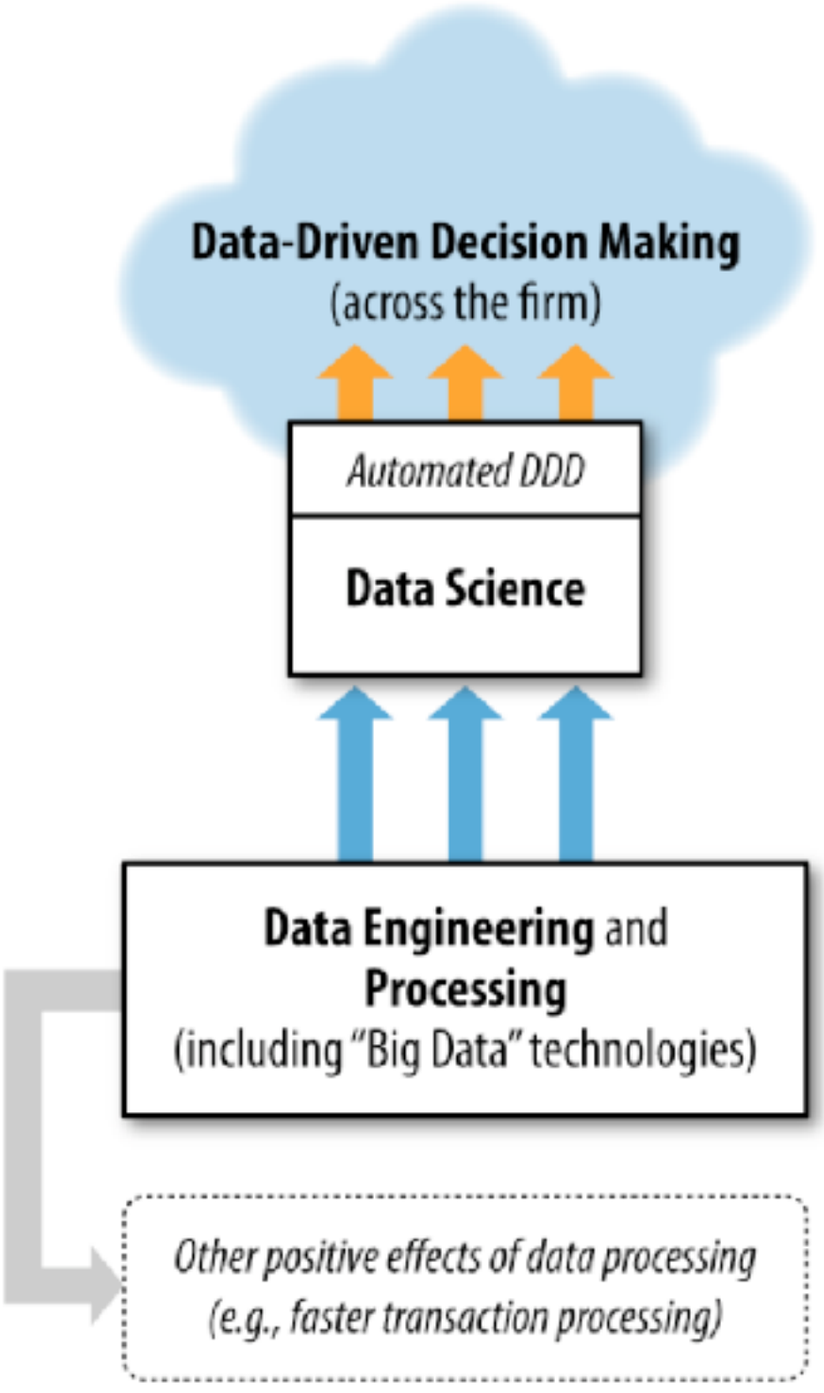
https://www.facebook.com/notes/mike-develin/debunking-princeton/10151947421191849


I USED TO THINK CORRELATION IMPLIED CAUSATION.
THEN I TOOK A STATISTICS CLASS. NOW I DON'T.
SOUNDS LIKE THE CLASS HELPED.
WELL, MAYBE.

Data Science and Data-Driven Decision Making

Introduction to Data Science

Data science in context of various data process in organication



DDD (Data-driven Decision Making)

- **Data-driven decision-making** (DDD) refers to the practice of basing decisions on the analysis of data, rather than purely on intuition
- DDD is not an all-or-nothing practice, and different firms engage in DDD to greater or lesser degrees.
- Economist Erik Brynjolfsson and his colleagues from MIT and Penn's Wharton School conducted a study of how DDD affects firm performance (Brynjolfsson, Hitt, & Kim, 2011). **They developed a measure of DDD that rates firms as to how strongly they use data to make decisions across the company**. **They show that statistically, the more data-driven a firm is, the more productive it is**—even controlling for a wide range of possible confounding factors. And the differences are not small. One standard deviation higher on the DDD scale is associated with a 4%–6% increase in productivity.

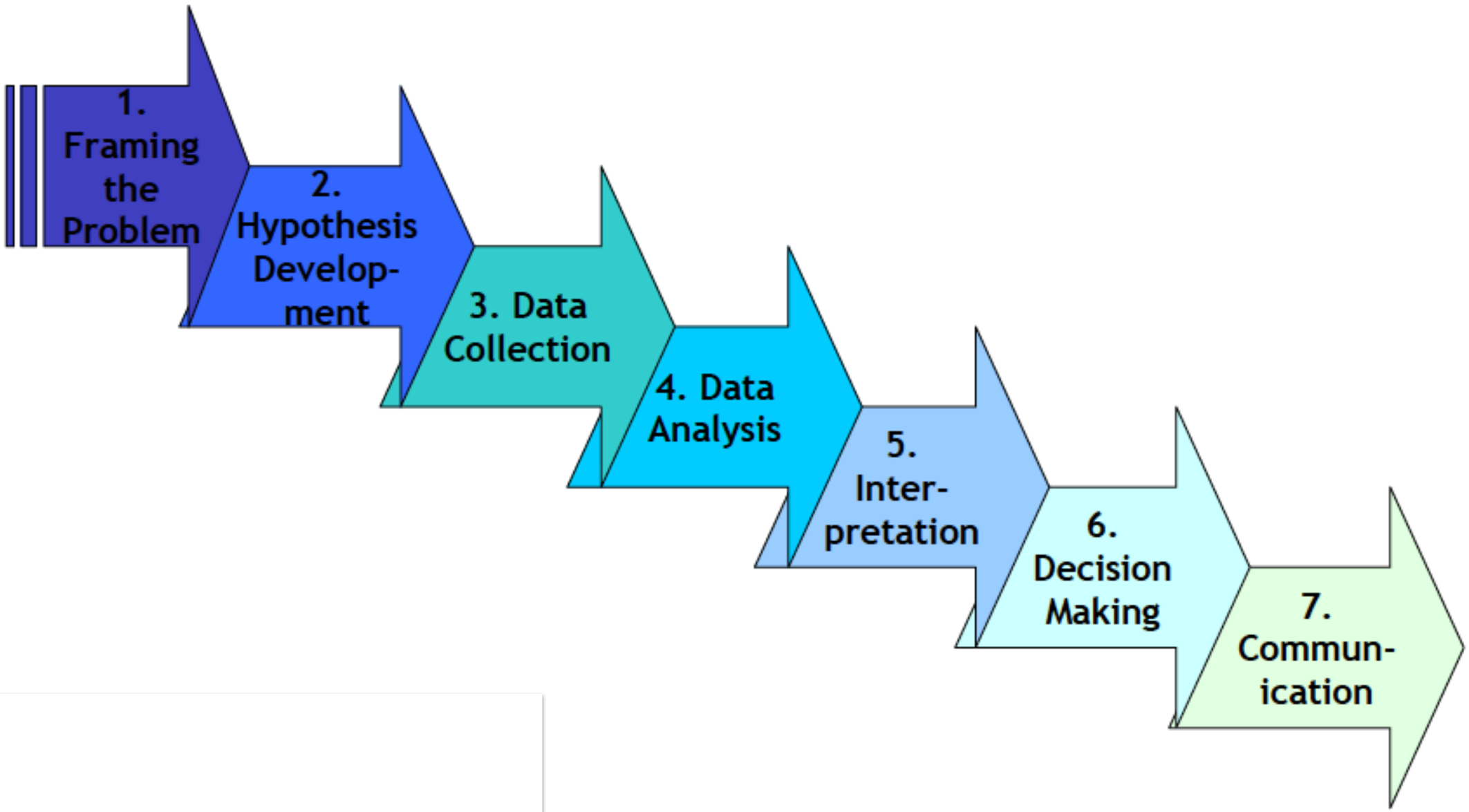
Decision type in Organization

1. Decisions for which “**discoveries**” need to be made within data
2. Decisions that repeat, especially at massive scale, and so decision-making can benefit from even small increases in **decision-making accuracy** based on data analysis.

Data and Data Science Capability as a Strategic Asset

- Achieving Competitive Advantage with Data Science
- Sustaining Competitive Advantage with Data Science

7 Stages of data-driven decision making process



» Where does your organization fall in analytic maturity?

Take the quiz!

1. How many data sources do you collect?

a. Why do we need a bunch of data?
– 0 points, end here.

b. I don't know the exact number.
– 5 points

c. We identified the required data and collect it. *– 10 points*

2. Do you know what questions your Data Science team is trying to answer?

a. Why do we need questions?
- 0 points

b. No, they figure it out for themselves.
– 5 points

c. Yes, we evaluated the questions that will have the largest impact to the business. *– 10 points*

3. Do you know the important factors driving your business?

a. I have no idea. *– 0 points*

b. Our quants help me figure it out.
– 5 points

c. We have a data product for that.
– 10 points

4. Do you have an understanding of future conditions?

a. I look at the current conditions and read the tea leaves. *– 0 points*

b. We have a data product for that.
– 5 points

5. Do you know the best course of action to take for your key decisions?

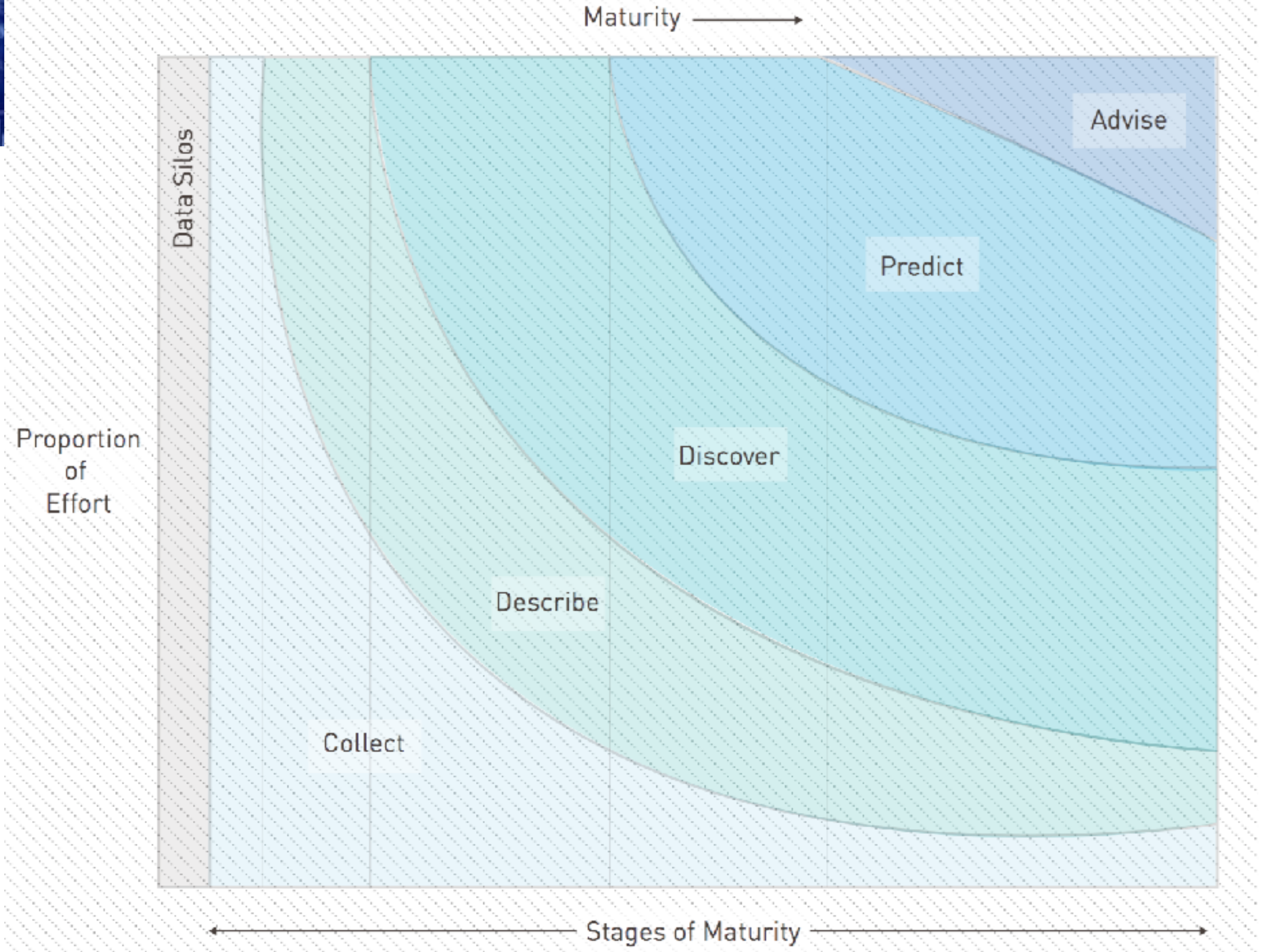
a. I look at the projections and plan a course. *– 0 points*

b. We have a data product for that.
– 5 points

Check your score:

0 – Data Silos, 5-10 – Collect,
10-20 – Describe, 20-30 – Discover,
30-35 – Predict, 35-40 - Advise

Source: Booz Allen Hamilton


Maturity
Data Silos
Advise
Predict
Discover
Describe
Collect
Proportion of Effort
Stages of Maturity

Analytic Maturity

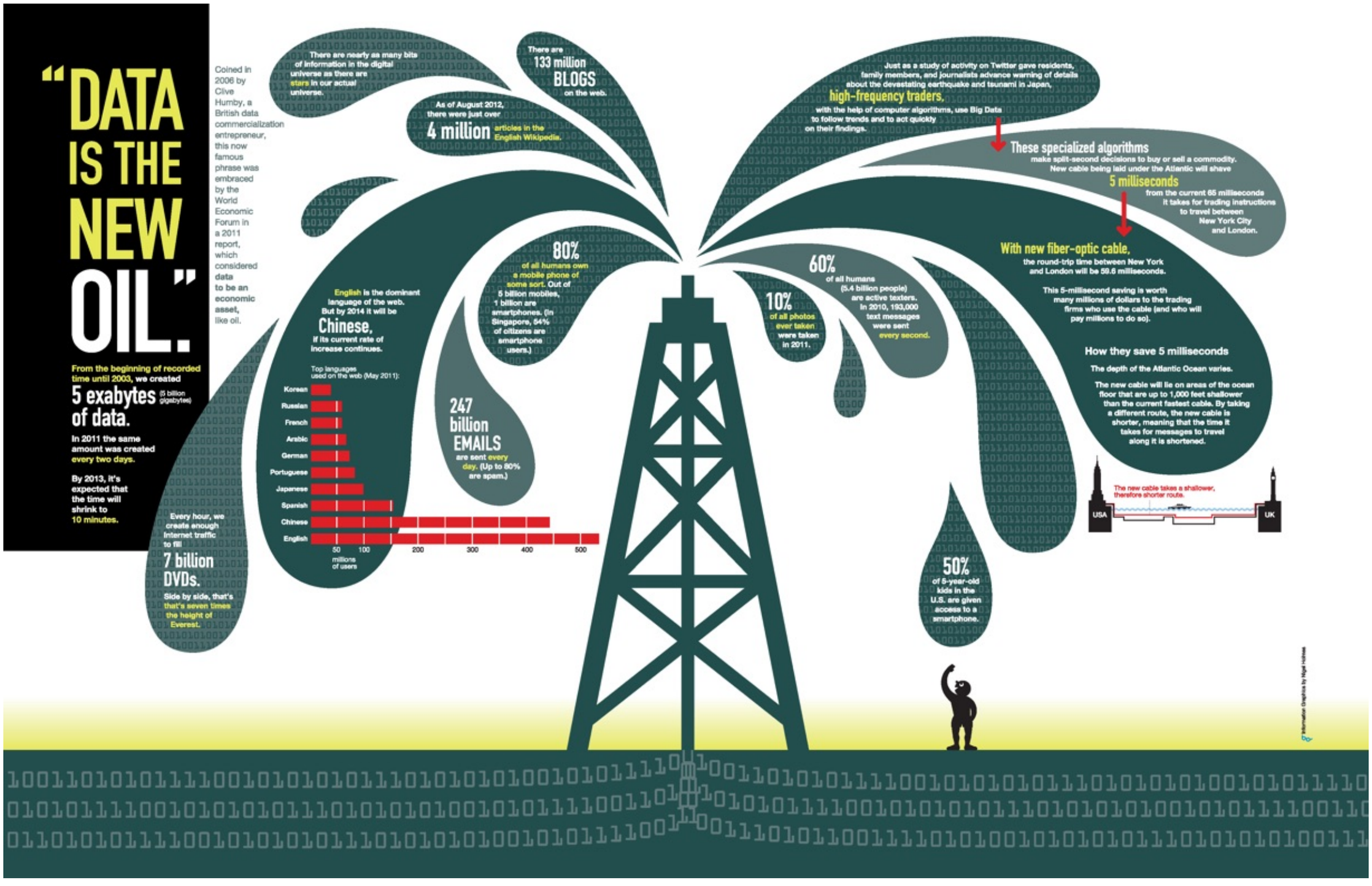
Stage	Description	Example
Collect	Focuses on collecting internal or external datasets.	Gathering sales records and corresponding weather data.
Describe	Seeks to enhance or refine raw data as well as leverage basic analytic functions such as counts.	How are my customers distributed with respect to location, namely zip code?
Discover	Identifies hidden relationships or patterns.	Are there groups within my regular customers that purchase similarly?
Predict	Utilizes past observations to predict future observations.	Can we predict which products that certain customer groups are more likely to purchase?
Advise	Defines your possible decisions, optimizes over those decisions, and advises to use the decision that gives the best outcome.	Your advice is to target advertise to specific groups for certain products to maximize revenue.


Regular Analytics
Big Data
Descriptive Analytics
What happened?
Diagnostic Analytics
Why did it happen?
Predictive Analytics
What will happen?
Prescriptive Analytics
How can we make it happen?
Information
Hindsight
Insight
Foresight
Optimization

Technology behind Data Science

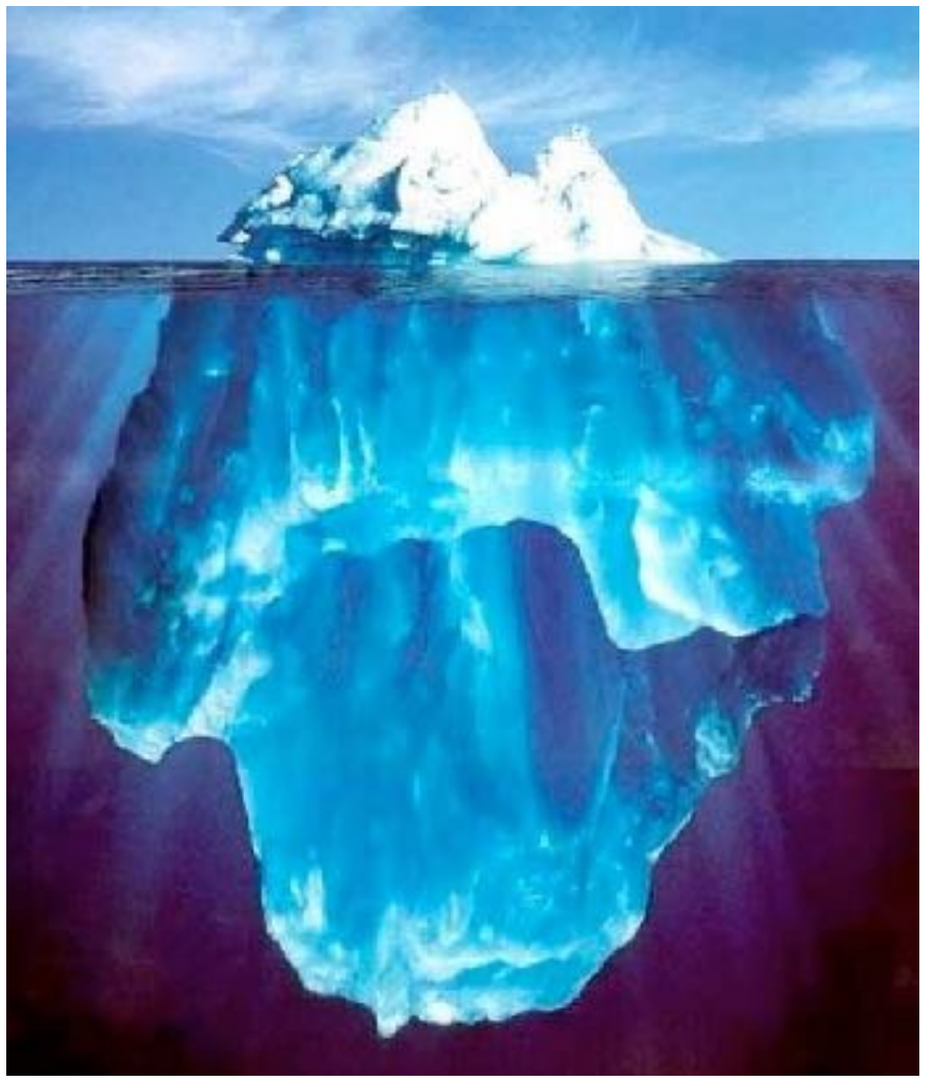
Data Science Certification Course

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



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
  - » …

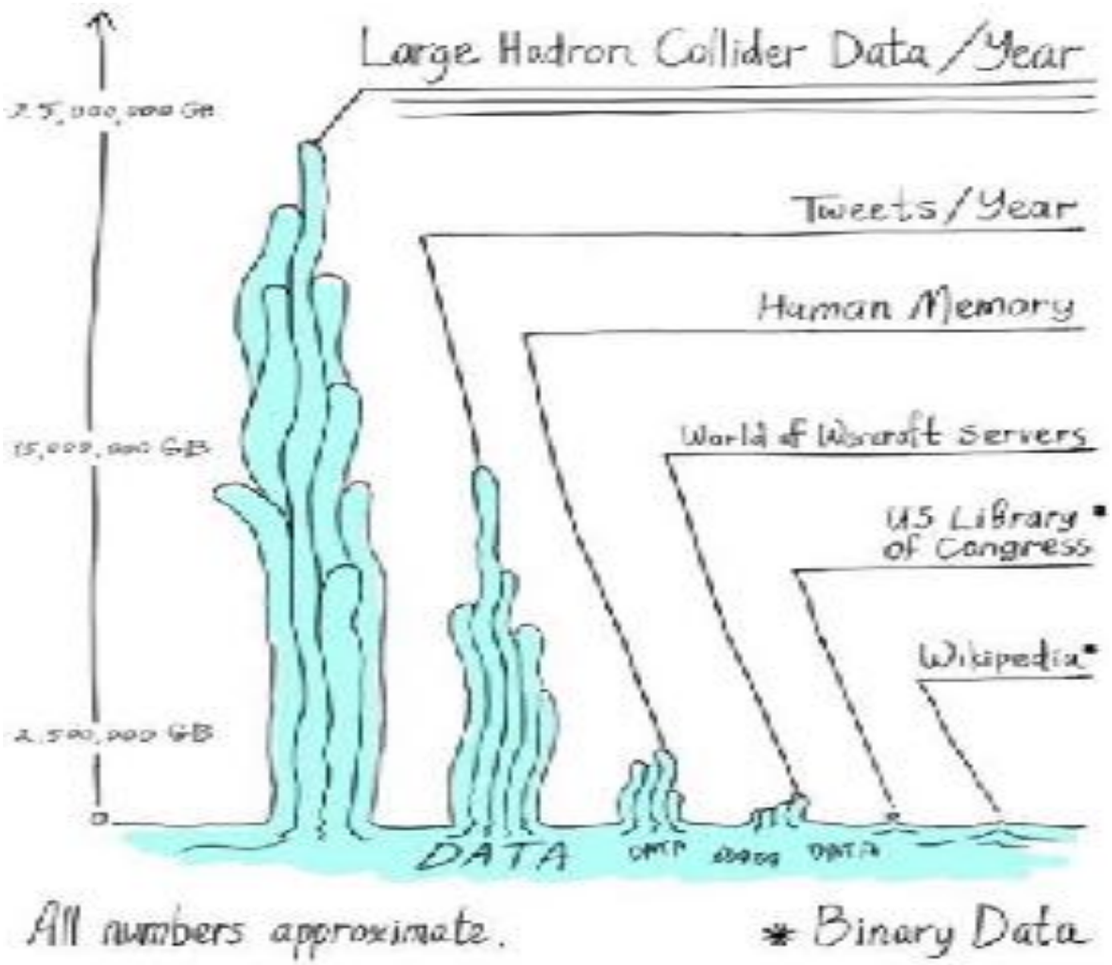


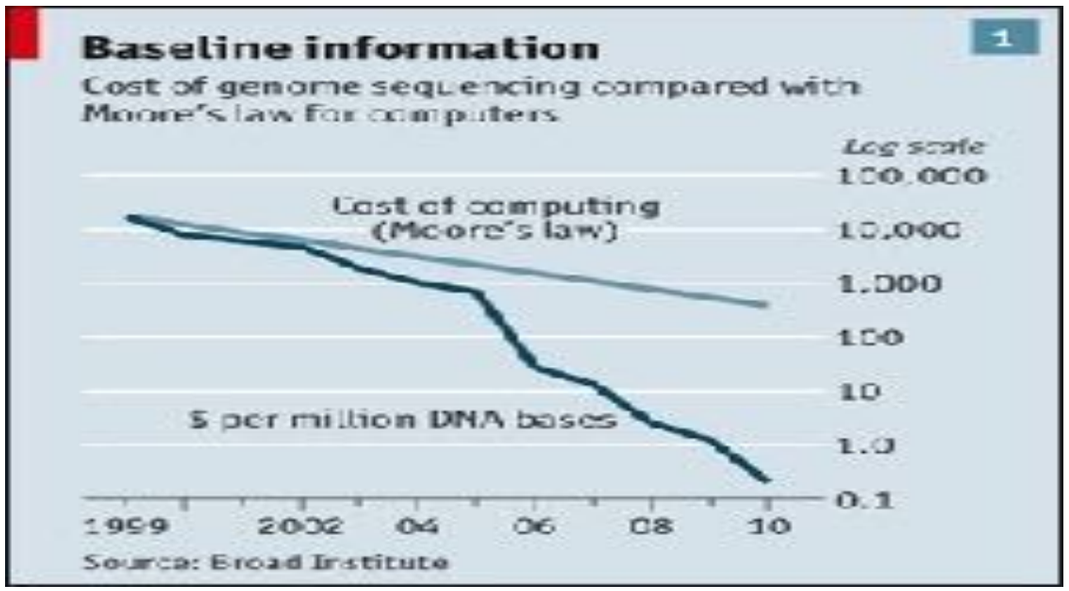
Where Does Big Data Come From?

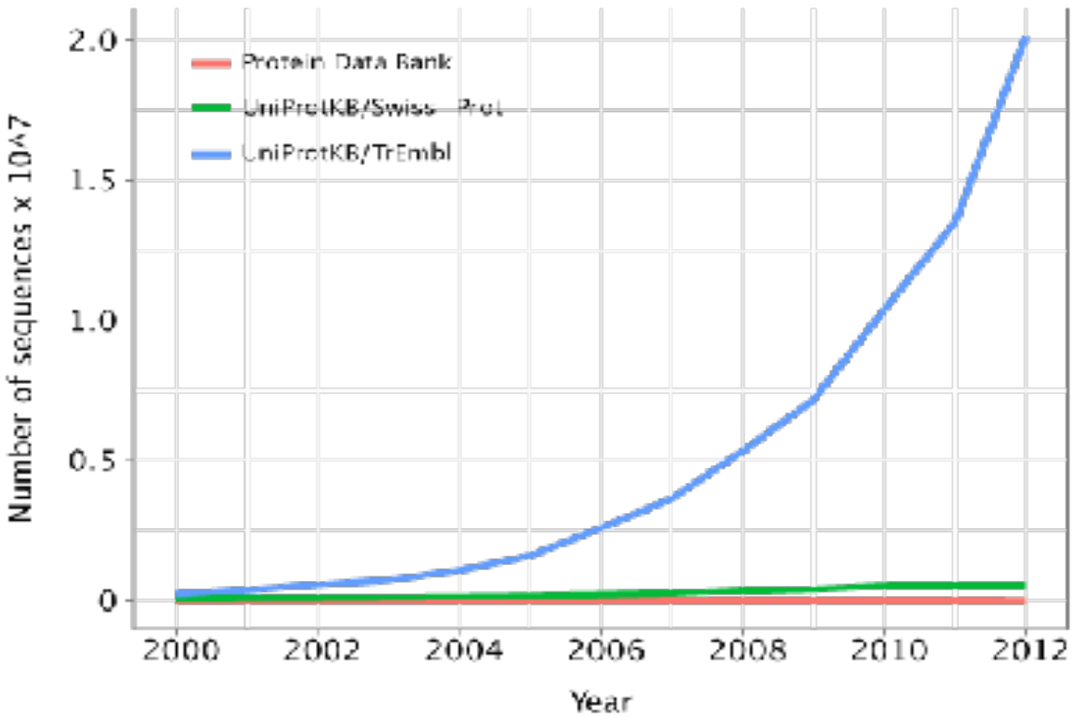
- User Generated Content (Web & Mobile)
  - » Facebook
  - » Instagram
  - » Yelp
  - » TripAdvisor
  - » Twitter
  - » YouTube
  - » …

Where Does Big Data Come From?

- Health and Scientific Computing





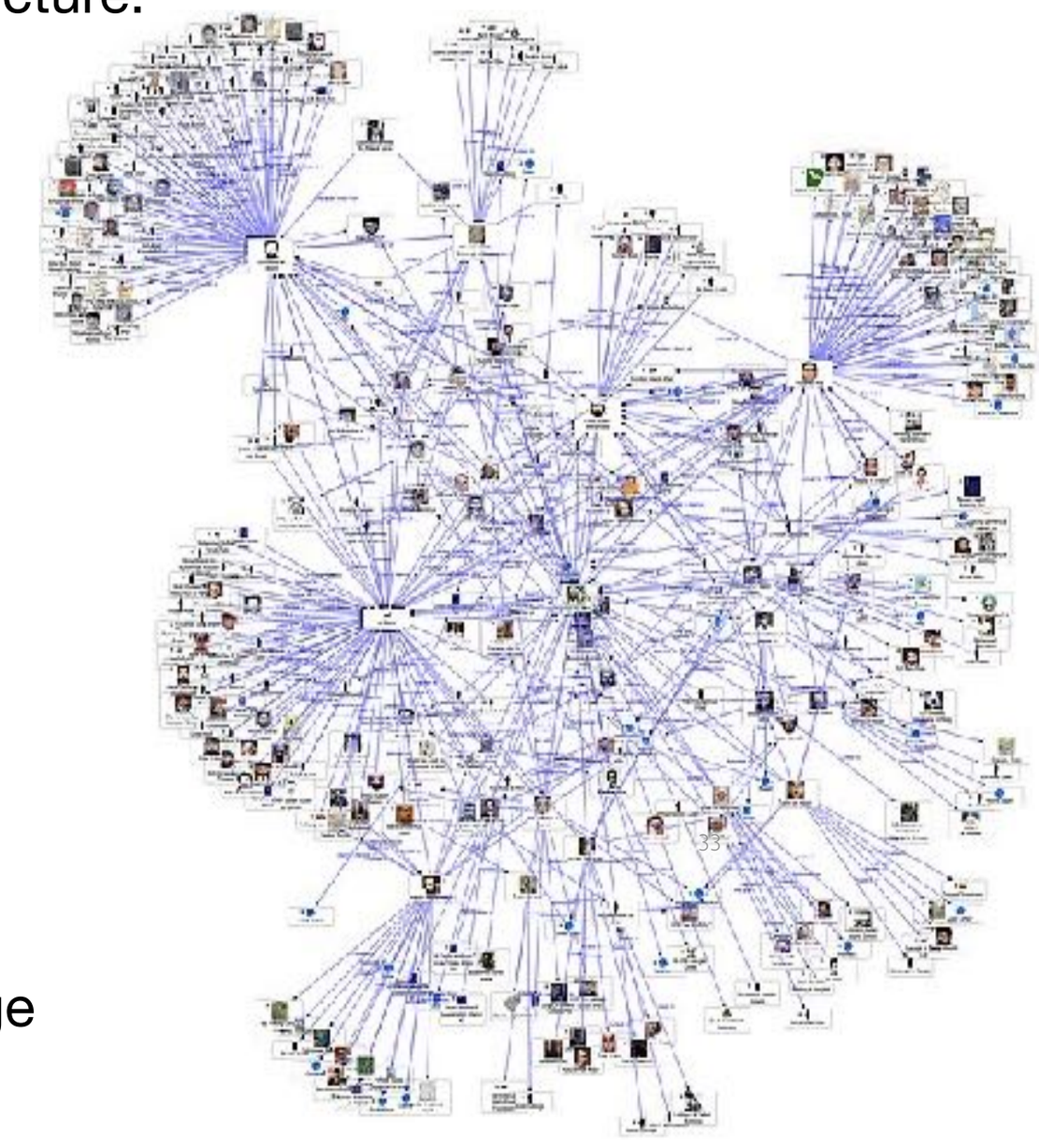


Images: http://www.economist.com/node/16349358 http://gorbi.irb.hr/en/method/growth-of-sequence-databases/ http://www.symmetrymagazine.org/article/august-2012/particle-physics-tames-big-data

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)

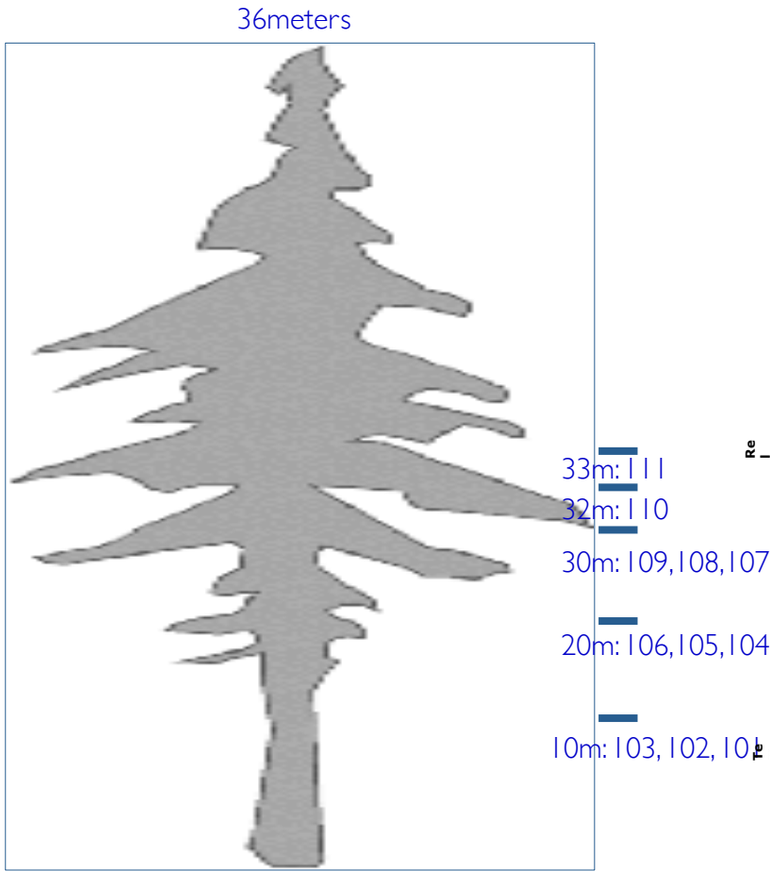
Log Files – Apache Web Server Log

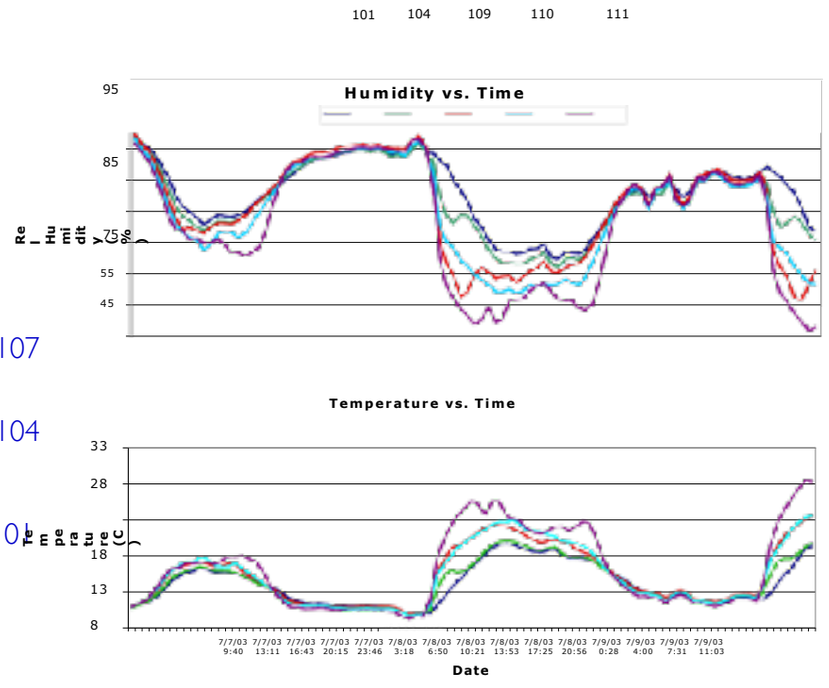
```
uplherc.upl.com . . [01/Aug/1995:00:00:07 .0400] "GET / HTTP/1.0" 304 0
uplherc.upl.com . . [01/Aug/1995:00:00:08 .0400] "GET /images/ksclogo.medium.gif
HTTP/1.0" 304 0
uplherc.upl.com . . [01/Aug/1995:00:00:08 .0400] "GET /images/MOSAIC.logosmall.gif
HTTP/1.0" 304 0
uplherc.upl.com . . [01/Aug/1995:00:00:08 .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:]:
                           ng.
unexpected field ID 6 with type 11. Skipping.
Feb 3 15:18:11 dhcp.47.129 Evernote[1140] <Warning>: . [EDAMAuthenticationResult read:]:
unexpected 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:]:
 unexpected 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





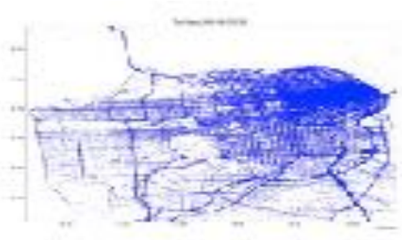
Redwood tree humidity and temperature at various heights

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

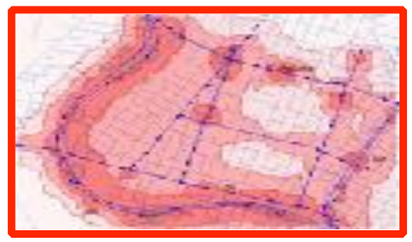
+

Physical modeling

+

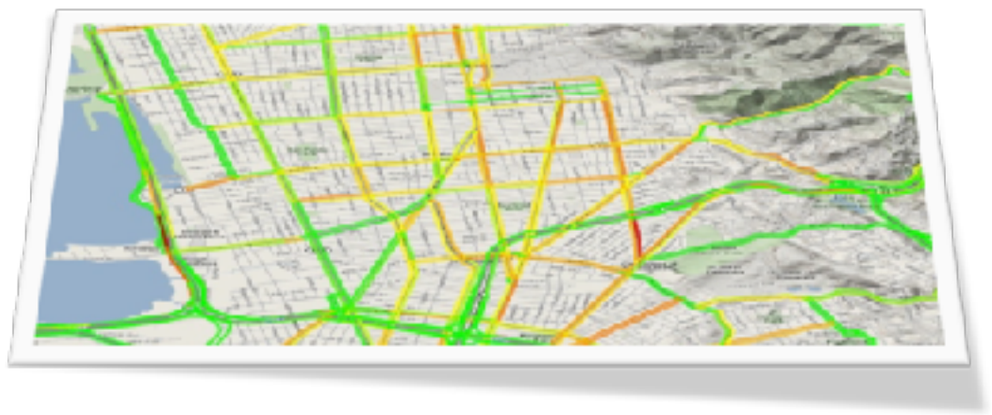
Sensing

+



Data Assimilation

=



http://traffic.berkeley.edu

Tools for Data Science Projects

DATA SOURCES

DATABASES

Structured

ORACLE
MySQL
SYBASE
memsql
TERADATA
Microsoft SQL Server

Unstructured

cloudera
Hortonworks
DATASTAX
hadapt
MAPR
mongoDB

APPLICATIONS

Sales

salesforce
SAP
RelateIQ
SUGARCRM

Marketing

Marketo
Hootsuite
MailChimp
KISSmetrics

Product

Google Analytics
Intercom
Optimizely
mixpanel

Customer

TOTANGO
Gainsight
zendesk
satisfaction

3RD PARTY DATA

Business Info

Experian
infogroup
D&B
THOMSON REUTERS
acxiom
factual

Social Media

GNIP
Digimind
sysomos
radian6
DATASIFT

Web Scrapers

import.io
kimonolabs
webhose.io
Connotate
dataprovider

Public Data

DATA.GOV
DataMarket
amazon web services
enigma

DATA WRANGLING

ENRICHMENT

Human

CrowdFlower
WorkFusion

Automated

tamr
TRIFACTA
Paxata
pentaho

ETL/BLENDING

etleap
CloverETL
alteryx
Astera

DATA INTEGRATION

snapLogic
databricks
ClearStory
informatica
apatar
talend
ZOOMDATA

API CONNECTORS

MuleSoft
Segment.io
zapier
IFTTT

DATA APPLICATIONS

INSIGHTS

Statistical Tools

SPSS
tableau
sas
MathWorks
STATA
Excel

Business Intelligence

GoodData
CHARTIO
DOMO
Qlik
birst
RJMETRICS
MicroStrategy
looker

Data Mining/ Exploration

Quid
Palantir
platfora
Datameer

Data Collaboration

MODE
Silk
Studio

MODELS

Predictive Analytics

Rapid Insight
Numenta
SALFORD SYSTEMS
SKYTREE

Natural Language Processing

LEXALYTICS
idibon
Maluuba
NUANCE
ATTENSITY
IBM WATSON

Deep Learning

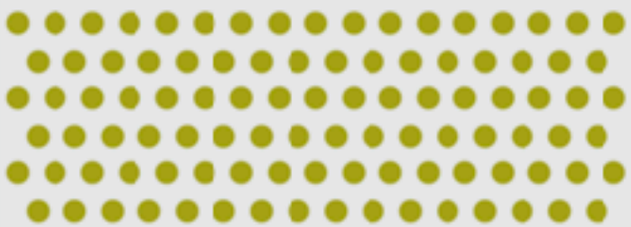
Dato
vicarious
MetaMind
AlchemyAPI
clarifai
SKYMIND

Machine Learning Platforms

kaggle
AzureML
Google

Big Data

Volume



Data at scale

Terabytes to petabytes of data

Variety

Data in many forms

Structured, unstructured, text, multimedia

Velocity

Data in motion

Analysis of streaming data to enable decisions within fractions of a second

Veracity

Data uncertainty

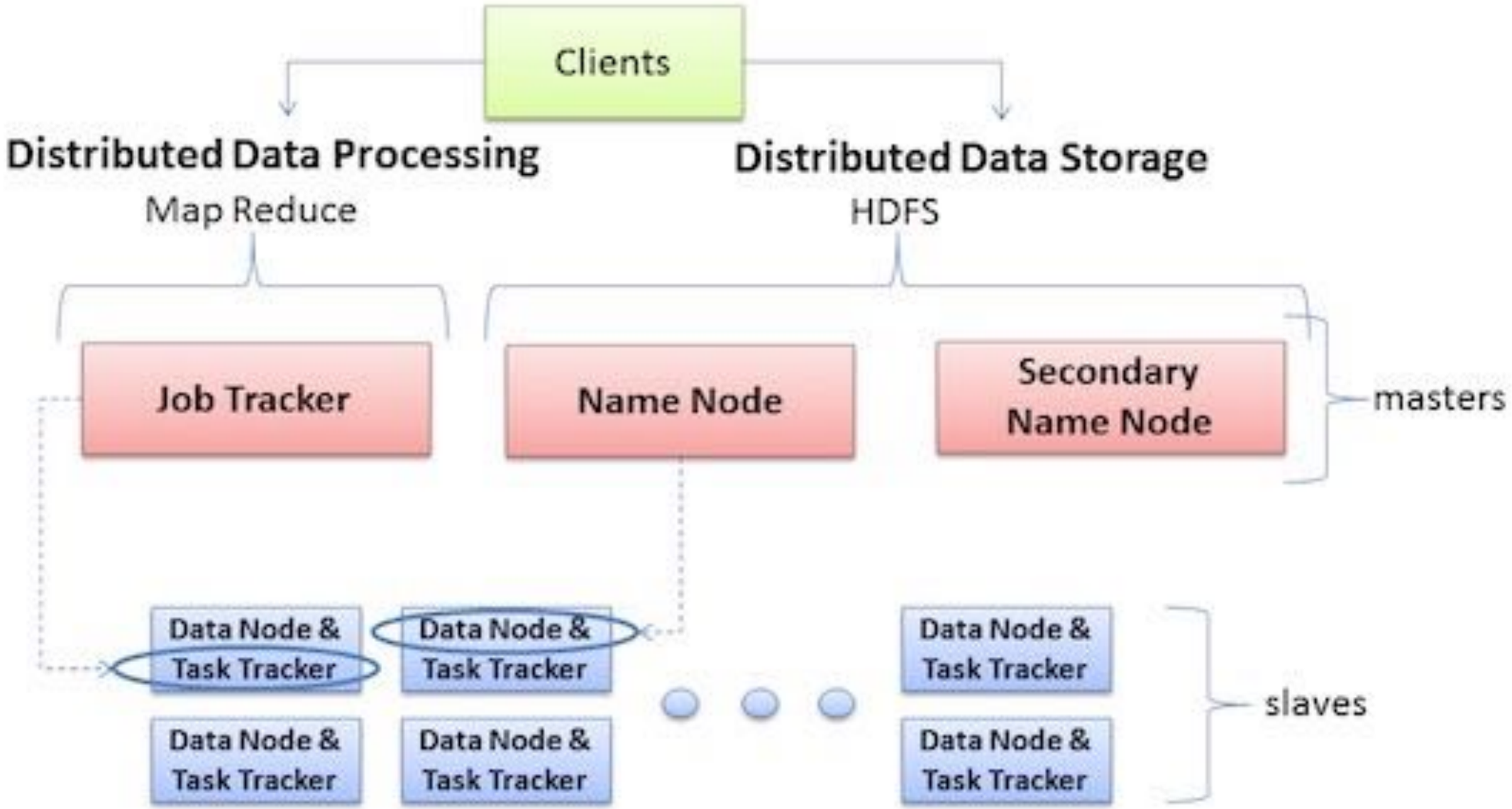
Managing the reliability and predictability of inherently imprecise data types

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

Data Science Certification Course

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.

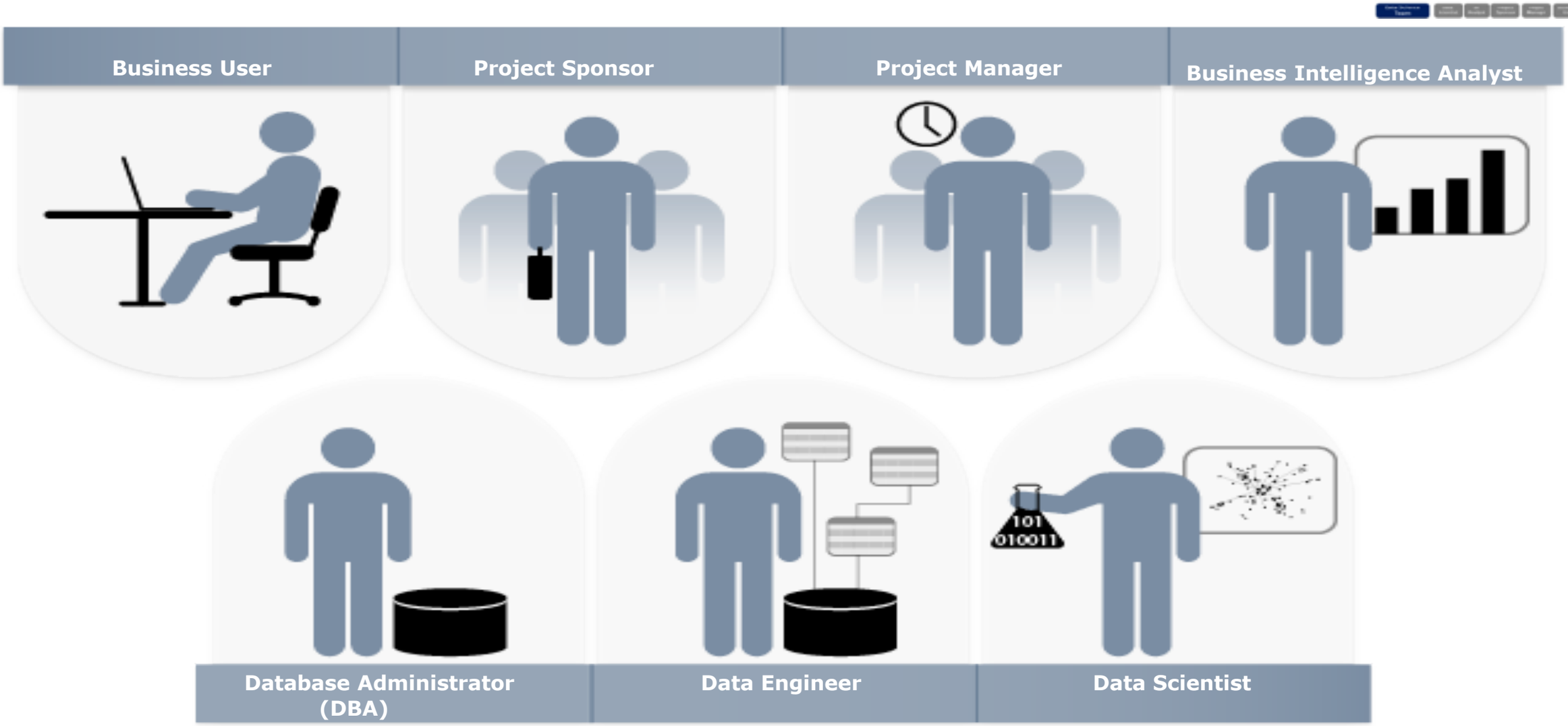
Hiring a unicorn?


HEY MIKE,
WHATS THE DIFFERENCE
BETWEEN A
DATA ANALYST
AND A
DATA SCIENTIST?
ABOUT 30k
PER ANNUM!
HA HA HA!
AM I RIGHT?
...MY NAME IS CLIVE

Setting up your Data Science Team

Data Science Certification Course

Successful Analytic Projects Require Breadth of Roles



Framework for Developing Data Science Teams

Developing Data Science Capabilities	Transforming	Creating	As-a-Service	Crowdsourcing

Data Science Team	Data Scientist	BI Analyst	Project Sponsor	Project Manager	Business User	Data Engineer	DBA

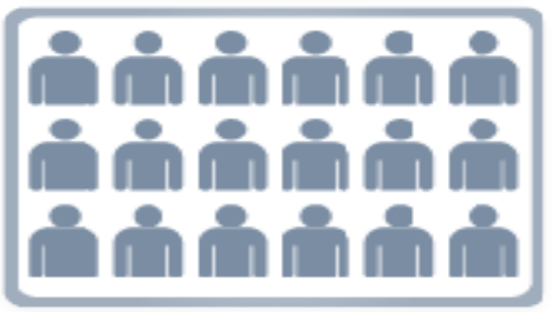
Four Approaches to Developing Data Science Capabilities

Transforming

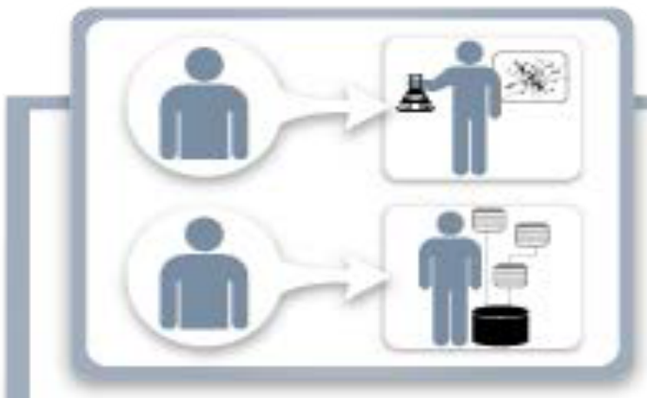
Creating

As a Service

Crowdsourcing



Approaches to Developing Data Science Capabilities:
Transforming Teams



Transforming And Realignment With Minimal Change To The Current Organizational Structure

- **Industries Requiring Deep Domain Knowledge** (Such As Genetics And DNA Sequencing)
- Established Companies Who Wish To Introduce Data Science Into Their Business
- Companies Who Wish To Enrich The In-house Skill Sets

Approaches to Developing Data Science Capabilities:
Transforming Teams

Developing A New Team From Scratch

- ***Start-up Companies***
- ***Companies Who Wish To …***
  - ***Increase Their Focus On Data Analytics***
  - ***Start New Data Science Projects***
- ***Companies Where Data Is The Product***
- ***Deep Domain Knowledge Is Less Critical For The Analytics***

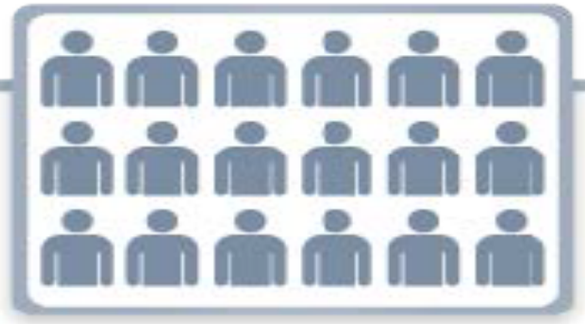
Approaches to Developing Data Science Capabilities:
Data Science as a Service

Engaging Data Science as a Service (DSaaS)

- ***When To Engage DSaaS Providers***
  - ***Prefer Not To Change Existing Organizational Structure***
  - ***When Creating Or Transforming Are Not Viable Options***

- ***Consider Service-level Agreements (SLAs) When Determining Whether To Engage Internal Resources Or External Providers***

Approaches to Developing Data Science Capabilities:
Crowdsourcing Data Science

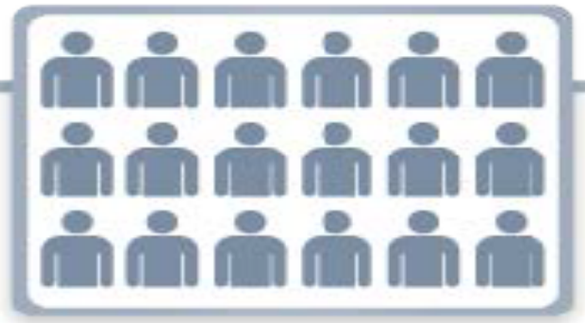


Outsource Data Science Project To Distributed Groups Of People

- ***When To Crowdsource***
  - ***The Problem Is "Open" In Nature***
  - ***Willing To Accept Opinions From Distributed And Diverse Groups Of People***
  - ***There's A Back-up Plan In Case Of "Crowd Failures"***

- ***Examples: Wikipedia, Netflix's $1,000,000 Prize***

Approaches to Developing Data Science Capabilities:
Crowdsourcing Data Science (Cont'd)



Outsource Data Science Projects To Distributed Groups Of People

- Different Crowdsourcing Models
  - Wisdom Of Crowds
  - Swarm Creativity (Collective Intelligence)
- Crowdsourcing Platforms
  - Kaggle.com, Innocentive.com
  - Amazon Mechanical Turk
- Crowd Failures:When The Turnout Of Crowdsourcing Is Unsatisfactory

Benefits and Drawbacks of the Four Approaches

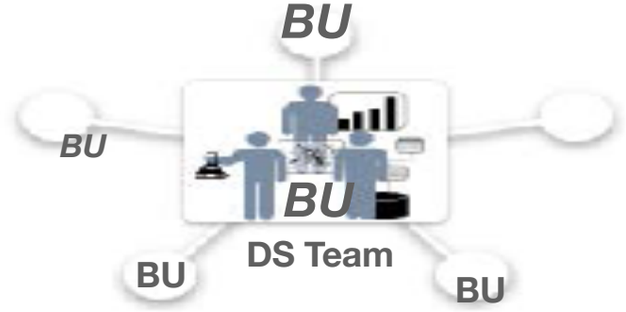
Transforming	Creating	DSaaS	Crowdsourcing
• **Strong Domain Knowledge** • Knowledge of Business Processes • New Talent Raises Level of Team Performance • Gradually Increases the Quality of Service	• **Control Over Skill-sets** • More Flexibility • High Quality of Service	• **Able to Scale on Demand** • May Get Better Service Levels Than In-house • Learn From Outside Experts	• **Leverage Wisdom of the Crowds** • Diverse Perspectives • Lower Cost • Fast Results
• **Risk of homogeneous thinking** • May Struggle With Quality of Service • Some Team Members May Resist Change	• **Hiring and Knowledge Transfer Are Time-consuming** • Time Required to Find and Hire Right Team Members	• **Provider May Not Understand Company's Unique Processes** • Difficult to Bring Expertise Back In-house • Decreasing Quality of Service Over Time	• **No SLA; value not guaranteed** • Difficult to design the "Open" Problem • Difficult For Domain Intensive Tasks • Crowd Failure May Happen (Adds Cost)

Framework for Developing Data Science Teams

Organizational Model	Centralized	Decentralized	Hybrid
Developing Data Science Capabilities	Transforming	Creating	As-a-Service	Crowdsourcing
Data Science Team	Data Scientist	BI Analyst	Project Sponsor	Project Manager	Business User	Data Engineer	DBA

Organizational Models for Data Science Teams

Centralized

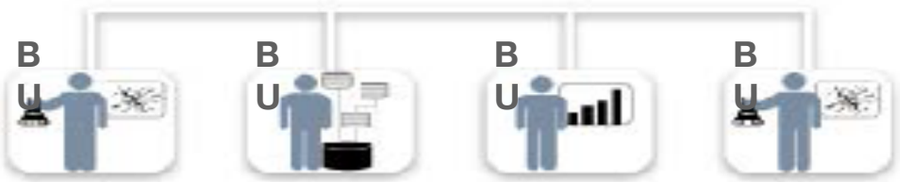


The Data Science Team Functions As A Hub And Spoke Model, In Which They Are A Central Provider Of Analytics To Multiple Business Units

Hybrid

There Is A Centralized Data Science Team, But Business Units Also Have Data Science Capabilities

Decentralized



Each Business Unit Has Its Own Data Science Capabilities

Regardless Of Which Approach, They All Need Executive Sponsorship To Succeed

Framework for Developing Data Science Teams

Executive Engagement	Data-driven CEO	Chief Data Officer					
Organizational Model	Centralized	Decentralized	Hybrid				
Developing Data Science Capabilities	Transforming	Creating	As-a-Service	Crowdsourcing			
Data Science Team	Data Scientist	BI Analyst	Project Sponsor	Project Manager	Business User	Data Engineer	DBA

Analytics Requires Executive Level Engagement

"Executive Sponsorship Is So Vital To Analytical Competition…"
-- Tom Davenport (Competing on Analytics)

Chief Strategy Officer
Simulate Outcomes for Acquiring Our Top 3 Competitors

Chief Product Officer
Conduct Social Media Analyses to Identify Customer Opinions

Chief Marketing Officer
Conduct Behavior Analyses to Predict If Customers Are Going to Churn

CEO

Executive Boardroom

Chief Finance Officer
Use Time Series Analysis Over Historical Data to Predict KPIs to Project Earnings

Chief Security Officer
Collect and Mine Log Data Within and Outside of the Company to Detect Unknown Threats

Chief Operating Officer
Mine Customer Opinions and Competitor Behaviors to Predict Inventory Demands

Executive Engagement: Data-Driven CEO

"… If Your Organization Can Arrange It … Have Someone In A Key Operational Role -- Business Unit Head, Chief Operations Officer, Even CEO -- To Be An Enthusiastic Advocate Of Matters Quantitative."

-- Tom Davenport (HBR Blog Network)

Key Focus Areas of a Data-driven CEO:

- ***Strategic Data Planning***
- ***Analytic Understanding***
- ***Technology Awareness***

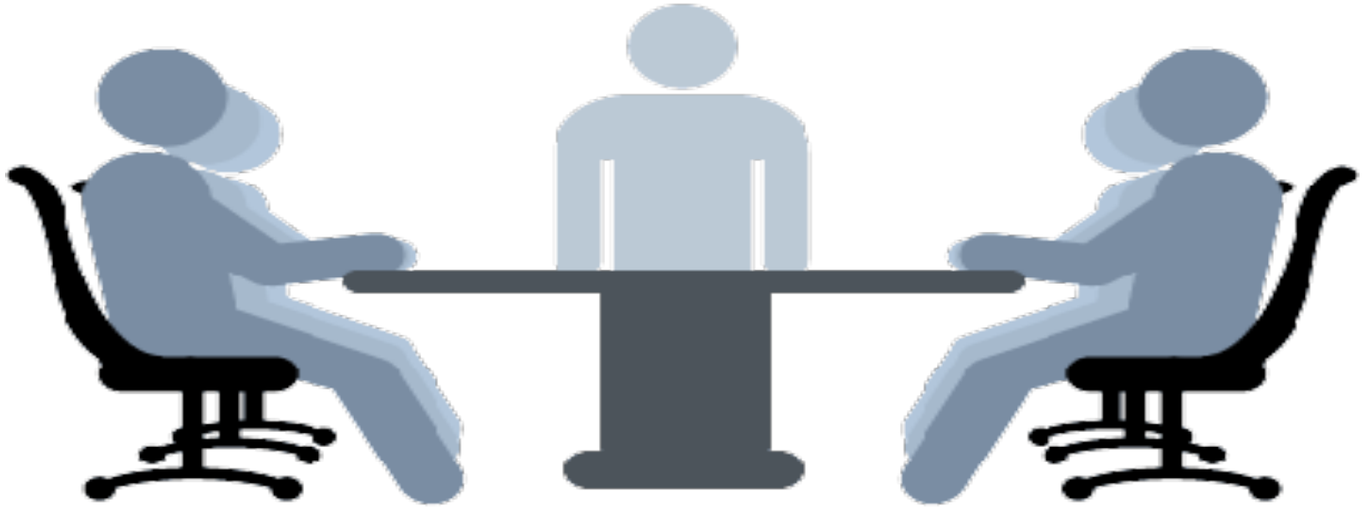
Procter & Gamble Business Sphere

Executive Engagement: Chief Data Officer (CDO)

> ***"… It's Time For Corporations To Embrace A New Functional Member Of The C-suite: The Chief Data Officer (CDO)."***
>
> *-- Anthony Goldbloom and Merav Bloch, Kaggle*

- Promote Data-driven Decision Making To Support Company's Key Initiatives
- Ensure The Company Collects The Right Data
- Oversee And Drive Analytics Company-wide

Executive-level Advisor On Data Analytics



Executive Boardroom

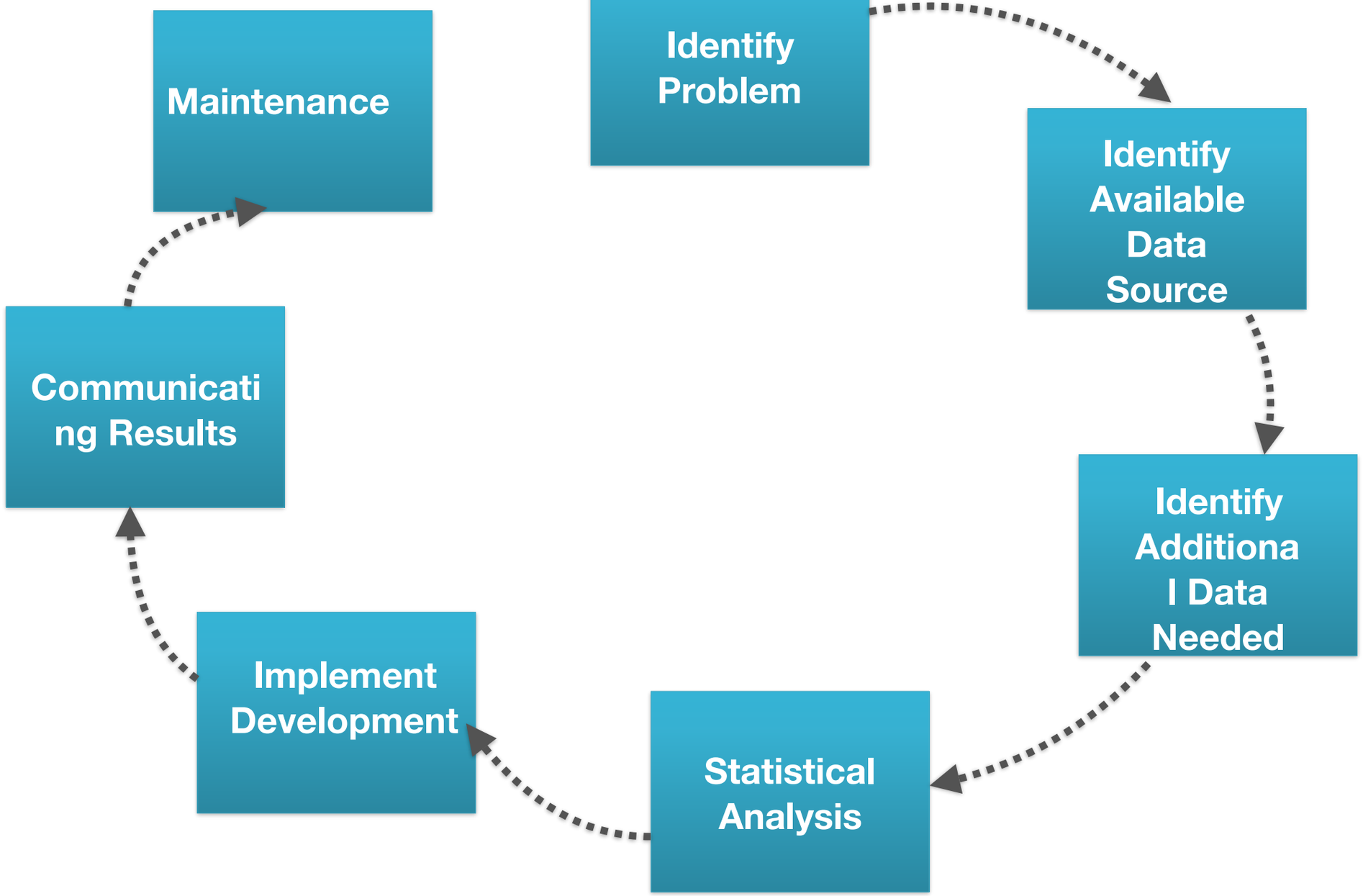
Data Science Life Cycle

Data Science Certification Course

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


Identify Problem
Identify Available Data Source
Identify Additional Data Needed
Statistical Analysis
Implement Development
Communicating Results
Maintenance

1. Identify the problem

- Identify metrics used to measure success over baseline (doing nothing)
- Identify type of problem: prototyping, proof of concept, root cause analysis, predictive analytics, prescriptive analytics, machine-to-machine implementation
- Identify key people within your organization and outside
- Get specifications, requirements, priorities, budgets
- How accurate the solution needs to be?
- Do we need all the data?
- Built internally versus using a vendor solution
- Vendor comparison, benchmarking

2. Identify available data sources

- Extract (or obtain) and check sample data (use sound sampling techniques); discuss fields to make sure data is understood by you
- Perform EDA (exploratory analysis, data dictionary)
- Assess quality of data, and value available in data
- Identify data glitches, find work-around
- Is quality and fields populated consistent over time?
- Are some fields a blend of different stuff (example: keyword field, sometimes equal to user query, sometimes to advertiser keyword, with no way to know except via statistical analyses or by talking to business people)
- How to improve data quality moving forward
- Do I need to create mini summary tables / database to
- Which tool do I need (R, Excel, Tableau, Python, Perl, SAS and so on)

3. Identify if additional data sources are needed

- What fields should be capture
- How granular
- How much historical data
- Do we need real time data
- How to store or access the data (NoSQL? Map-Reduce?)
- Do we need experimental design?

4. Statistical Analyses

- Use imputation methods as needed
- Detect / remove outliers
- Selecting variables (variables reduction)
- Is the data censored (hidden data, as in survival analysis or time-to-crime statistics)
- Cross-correlation analysis
- Model selection (as needed, favor simple models)
- Sensitivity analysis
- Cross-validation, model fitting
- Measure accuracy, provide confidence intervals

5. Implementation, development

- FSSRR: Fast, simple, scalable, robust, re-usable
- How frequently do I need to update lookup tables, white lists, data uploads, and so on
- Debugging
- Need to create an API to communicate with other apps?

6. Communicate results

- Need to integrate results in dashboard? Need to create an email alert system?
- Decide on dashboard architecture, with business people
- Visualization
- Discuss potential improvements (with cost estimates)
- Provide training
- Commenting code, writing a technical report, explaining how your solution should be used, parameters fine-tuned, and results interpreted

7. Maintenance

- Test the model or implementation; stress tests
- Regular updates
- Final outsourcing to engineering and business people in your company, once solutions is stable
- Help move solution to new platform or vendor

Real Data Science

Use Cases

Data Science Certification Course

Background

For each type of data science case think about

1) What problem does it solve, and for whom?

2) How is it being solved today?

3) How can it beneficially affect business?

4) What are the data inputs and where do they come from?

5) What are the outputs and how are they consumed (online algorithm, a static report etc)

6) Is this cost saving or revenue growth problem?

By function : Marketing

1) Predicting Lifetime Value (LTV)

what for: if you can predict the characteristics of high LTV customers, this supports customer segmentation, identifies upsell opportunities and supports other marketing initiatives

usage: can be both an online algorithm and a static report showing the characteristics of high LTV customers

By function : Marketing

2) Wallet share estimation

working out the proportion of a customer's spend in a category accrues to a company allows that company to identify upsell and cross-sell opportunities

usage: can be both an online algorithm and a static report showing the characteristics of low wallet share customers

3) Churn

working out the characteristics of churners allows a company to product adjustments and an online algorithm allows them to reach out to churners

usage: can be both an online algorithm and a statistic report showing the characteristics of likely churners

By function : Marketing

4) Customer segmentation

If you can understand qualitatively different customer groups, then we can give them different treatments (perhaps even by different groups in the company). Answers questions like: what makes people buy, stop buying etc

usage: static report

5) Product mix

What mix of products offers the lowest churn? eg. Giving a combined policy discount for home + auto = low churn

usage: online algorithm and static report

By function : Marketing

6) Cross selling/Recommendation algorithms/

Given a customer's past browsing history, purchase history and other characteristics, what are they likely to want to purchase in the future?

usage: online algorithm

7) Up selling

Given a customer's characteristics, what is the likelihood that they'll upgrade in the future?

usage: online algorithm and static report

By function : Marketing

8) Channel optimization

what is the optimal way to reach a customer with certain characteristics?

usage: online algorithm and static report

9) Discount targeting

What is the probability of inducing the desired behavior with a discount

usage: online algorithm and static report

10) Reactivation likelihood

What is the reactivation likelihood for a given customer

usage: online algorithm and static report

11) Adwords optimization and ad buying

calculating the right price for different keywords/ad slots

By function : Marketing

12) Target market

Understanding the target helps you determine exactly what your products or services will be, and what kind of customer service tactics work best

usage: static report

By function: Sales

1) Lead prioritization

What is a given lead's likelihood of closing

r**evenue impact:** supports growth

usage: online algorithm and static report

2) Demand forecasting

What product will customer buy next?

By function : Risk

1) Credit risk

Treasury or currency risk

How much capital do we need on hand to meet these requirements?

2) Fraud detection

predicting whether or not a transaction should be blocked because it involves some kind of fraud (eg credit card fraud)

3) Accounts Payable Recovery

Predicting the probably a liability can be recovered given the characteristics of the borrower and the loan

4) Anti-money laundering

Using machine learning and fuzzy matching to detect transactions that contradict AML legislation (such as the OFAC list)

By function : Customer support

1) Call centers

Call routing (ie determining wait times) based on caller id history, time of day, call volumes, products owned, churn risk, LTV, etc.

2) Call center message optimization

Putting the right data on the operator's screen

3) Call center volume forecasting

predicting call volume for the purposes of staff rostering

By function : Human Resources

1) Resume screening

scores resumes based on the outcomes of past job interviews and hires

2) Employee churn

predicts which employees are most likely to leave

3**) Training recommendation**

recommends specific training based of performance review data

4) Talent management

looking at objective measures of employee success

By function : Operation

1) Detecting employee

tracking work hours or detecting employee thieves

Telecom industries

- Dropped calls
- Lack of network coverage, resulting in poor customer experience
- Bandwidth issues
- Poor download times
- Inordinate service wait times
- Poorly trained customer service reps
- Inadequate call center staffing

Healthcare

- **Claims review prioritization**
    - Payers picking which claims should be reviewed by manual auditors
- **Medicare/medicaid fraud**
    - Tackled at the claims processors, EDS is the biggest & uses proprietary tech
- **Medical resources allocation**
    - Hospital operations management
    - Optimize/predict operating theatre & bed occupancy based on initial patient visits

Healthcare (Continued)

- **Alerting and diagnostics from real-time patient data**
- **Prescription compliance**
  - Predicting who won't comply with their prescriptions
- **Physician attrition**
  - Hospitals want to retain Drs who have admitting privileges in multiple hospitals
- **Survival analysis**
  - Analyze survival statistics for different patient attributes (age, blood type, gender, etc) and treatment

Healthcare (Continued)

- **Medication (dosage) effectiveness**
    - Analyze effects of admitting different types and dosage of medication for a disease
- **Readmission risk**
    - Predict risk of re-admittance based on patient attributes, medical history, diagnose & treatment

Retail

Pricing

Optimize per time period, per item, per store

Location of new stores

Product layout in stores

Merchandizing

when to start stocking & discontinuing product lines

Inventory Management (how many units)

In particular, perishable goods

Shrinkage analytics

Theft analytics/prevention

Retail

Warranty Analytics

Rates of failure for different components

And what are the drivers or parts?

What types of customers buying what types of products are likely to actually redeem a warranty?

Market Basket Analysis

Cannibalization Analysis

Insurance

Claims prediction

Might have telemetry data

Claims handling (accept/deny/audit), managing repairer network (auto body, doctors)

Price sensitivity

Agent & branch performance

Product mix

Manufacturing

Predictive Maintenance

Sensor data to look at failures

Quality management

Identifying out-of-bounds manufacturing

Visual inspection/computer vision

Optimal run speeds

Demand forecasting/inventory management

Warranty/pricing

Agriculture

Yield management (taking sensor data on soil quality)

Travel

- Aircraft scheduling
- Seat management, gate management
- Air crew scheduling
- Dynamic pricing
- Customer complain resolution (give points in exchange)
- Tourism forecasting

Workshop 1 - Find your data science questions

- Each person will pick the questions that your business want you to answer. List them in papers and Exchange to the person next to you to criticize. Then you have one chance to modify them.
- Present them to the class

R Programming

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 **R**oss Ihaka and **R**obert 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

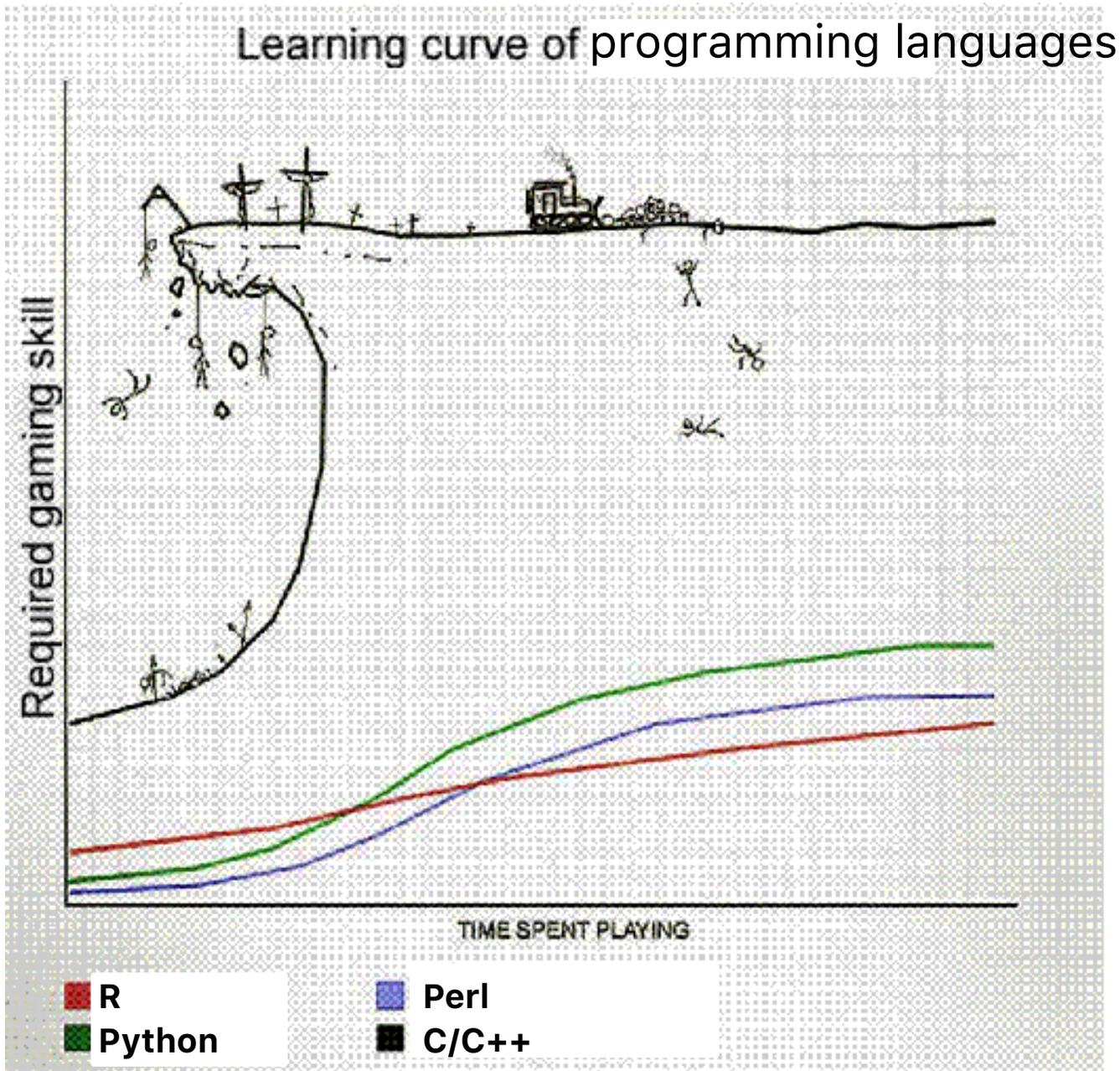
What is R?

- 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 10,000+ different user-–written libraries available (http://cran.r-project.org)

Why use R?

- R is Open-Source Software
- R has been used for Statistical Computing for more than 20 years
- 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?



http://www.vayapotra.es/wordpress/wp-content/uploads/2009/02/2rmqi6

Data Science Programming Language

Platform	2019 % share	2018 % share	% change
Python	65.8%	65.6%	0.2%
R Language	46.6%	48.5%	-4.0%
SQL Language	32.8%	39.6%	-17.2%
Java	12.4%	15.1%	-17.7%
Unix shell/awk	7.9%	9.2%	-13.4%
C/C++	7.1%	6.8%	3.7%
Javascript	6.8%	na	na
Other programming and data languages	5.7%	6.9%	-17.1%
Scala	3.5%	5.9%	-41.0%
Julia	1.7%	0.7%	150.4%
Perl	1.3%	1.0%	25.2%
Lisp	0.4%	0.3%	46.1%

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.*

Installing R and R-Studio

Download R

https://www.r-project.org

[Home]

Download

CRAN

R Project

About R
Logo
Contributors
What's New?
Reporting Bugs
Conferences
Search
Get Involved: Mailing Lists
Developer Pages
R Blog

R Foundation

The R Project for Statistical Computing

Getting Started

R is a free software environment for statistical computing and graphics. It compiles and runs on a wide variety of UNIX platforms, Windows and MacOS. To **download R**, please choose your preferred CRAN mirror.

If you have questions about R like how to download and install the software, or what the license terms are, please read our answers to frequently asked questions before you send an email.

News

- **R version 3.6.1 (Action of the Toes)** has been released on 2019-07-05.
- useR! 2020 will take place in St. Louis, Missouri, USA.
- **R version 3.5.3 (Great Truth)** has been released on 2019-03-11.
- The R Foundation Conference Committee has released a call for proposals to host useR! 2020 in North America.

Download R Studio

https://rstudio.com/products/rstudio/download/#download

	RStudio Desktop Open Source License **Free** DOWNLOAD Learn more	RStudio Desktop Commercial License **$995** /year BUY Learn more	RStudio Server Open Source License **Free** DOWNLOAD Learn more	RStudio Server Pro Commercial License **$4,975** /year (5 Named Users) BUY Evaluation | Learn more
Integrated Tools for R	✓	✓	✓	✓
Priority Support		✓		✓
Access via Web Browser			✓	✓
Enterprise Security				✓
Project Sharing				✓
Manage Multiple R Sessions & Versions				✓
Admin Dashboard				✓
Load Balancing				✓
Auditing and Monitoring				✓
Data Connectivity				✓
Launcher				✓
Tutorial API				✓
License	AGPL	Commercial	AGPL	Commercial

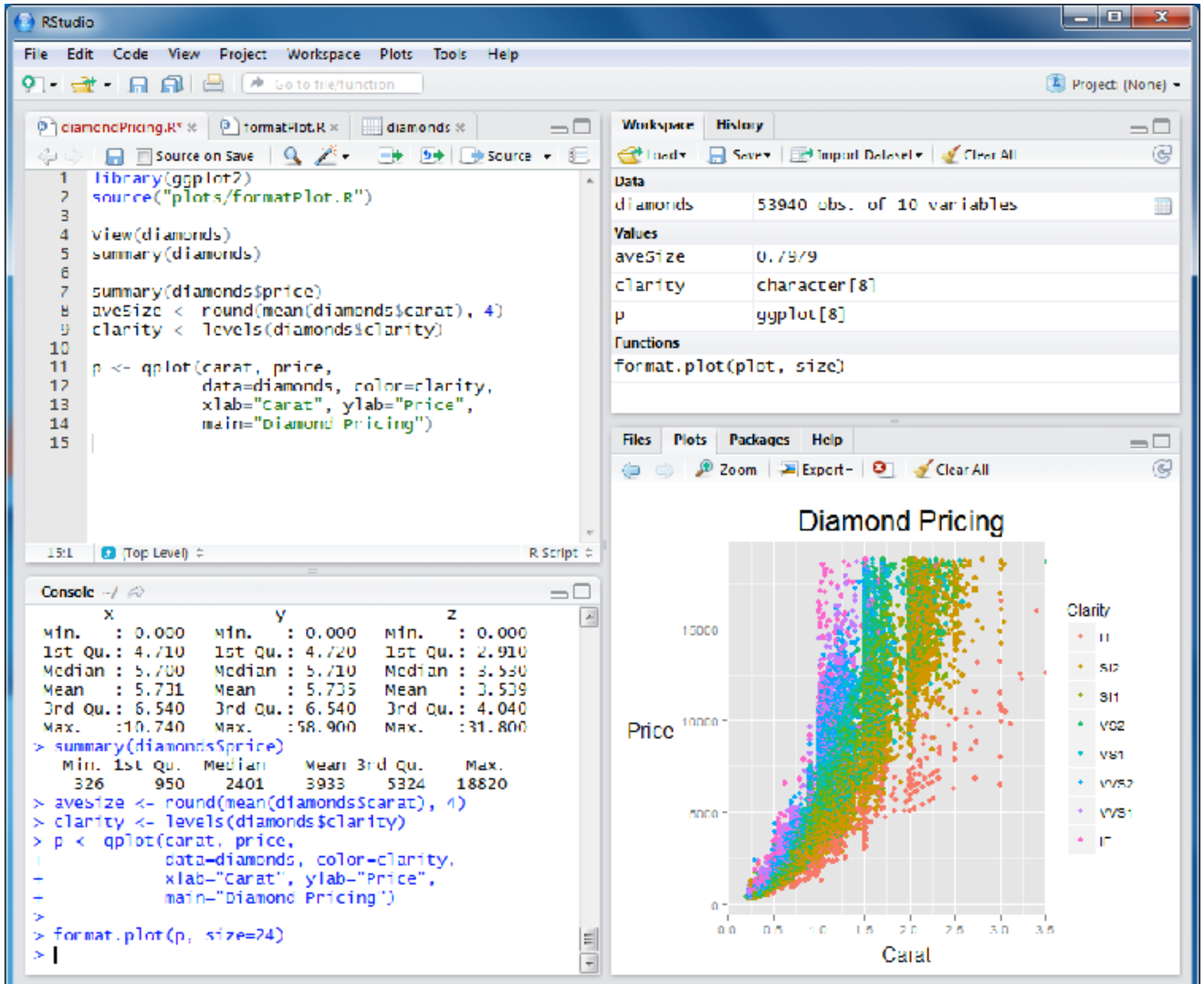
Installers for Supported Platforms

Installers	Size	Date	MD5
RStudio 1.2.5019 - Ubuntu 18/Debian 10 (64-bit)	106.04 MB	2019-11-01	a6c9af3d8b1621eb155d23c879c1a75a
RStudio 1.2.5019 - Debian 9 (64-bit)	106.39 MB	2019-11-01	bc7b0b25b41e39fb6f1aefa74163a133
RStudio 1.2.5019 - Fedora 28/Red Hat 8 (64-bit)	120.89 MB	2019-11-01	2291b1befb02622b3aa02c43638ee5c2
RStudio 1.2.5019 - macOS 10.12+ (64-bit)	126.88 MB	2019-11-01	55738355277e8ec660e628acaf2a401b
RStudio 1.2.5019 - SLES/OpenSUSE 12 (64-bit)	99.04 MB	2019-11-01	3bcbf47f40944cc4a5ef4f6fb42319c1
RStudio 1.2.5019 - OpenSUSE 15 (64-bit)	107.09 MB	2019-11-01	29d07b198b7aac92356f8487911efbfa
RStudio 1.2.5019 - Fedora 19/Red Hat 7 (64-bit)	120.26 MB	2019-11-01	dab1cb5f0ed39f5bcf0c795e2938fa94
RStudio 1.2.5019 - Ubuntu 14/Debian 8 (64-bit)	96.93 MB	2019-11-01	f86811fce50b48850fed259d6ce7ef13
RStudio 1.2.5019 - Windows 10/8/7 (64-bit)	149.82 MB	2019-11-01	4d6521a9b89d70c3bf50414c8b6708f2
RStudio 1.2.5019 - Ubuntu 16 (64-bit)	104.91 MB	2019-11-01	67d5a2c255f2bc1a171c7e417853102c

RStudio Features

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

Introducing R-Studio



RStudio Panes

RStudio has 4 main windows ('panes'):

- **The Source pane**: create a file that you can save and run later
- **The Console pane**: type or paste in commands to get output from R
- **The Workspace/History pane**: see a list of variables or previous commands
- **The Files/Plots/Packages/Help pane**: see plots, help pages, and other items in this window.

Source and Console Panes

Source window: Create a file here, so that you can save and run it later (or turn in as homework)

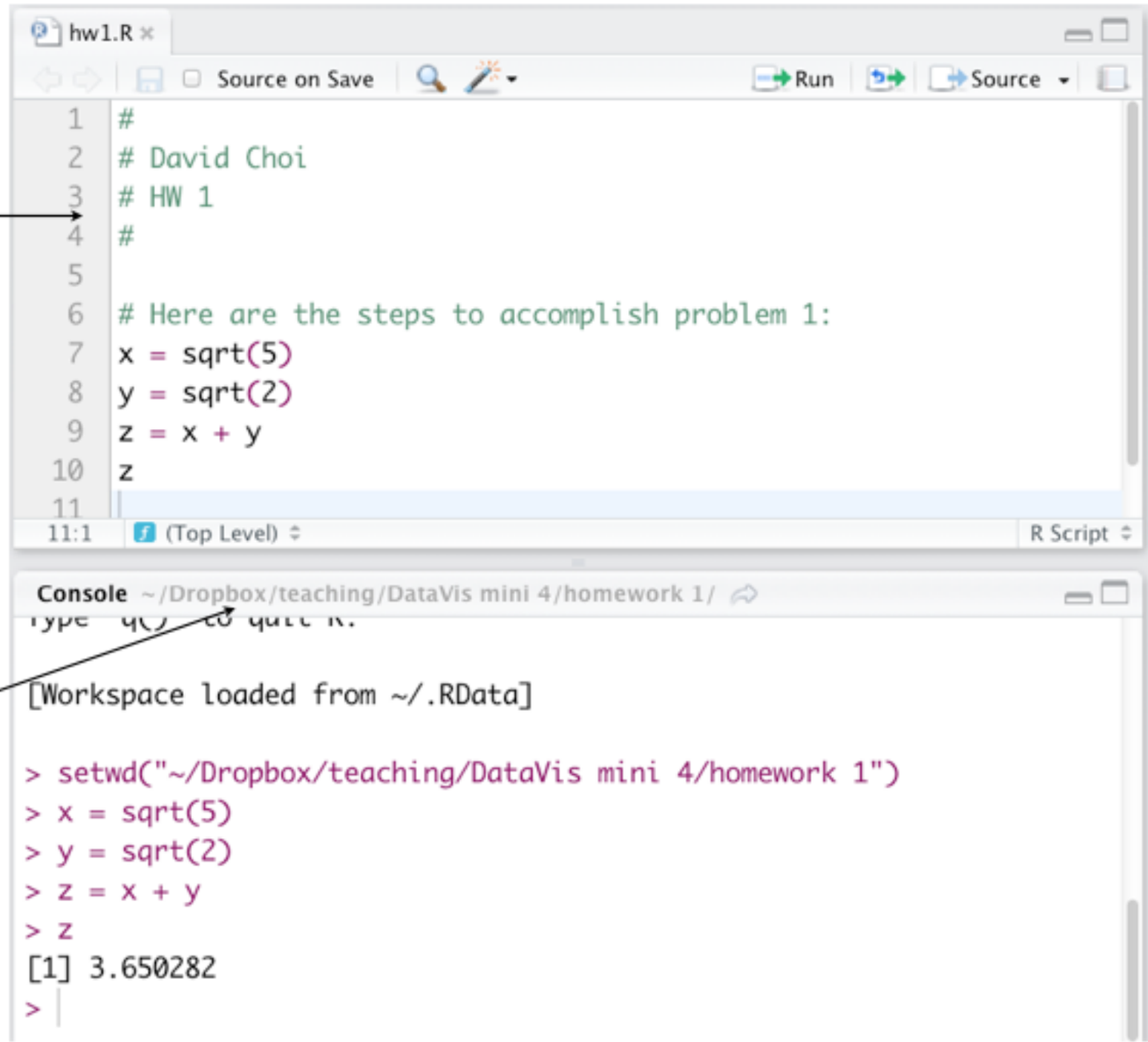
Any non-command line should start with a #

Console: Type or paste commands in here to get results from R

If you are loading a data file, you will need to be in the correct directory

> denotes that R is waiting for a command

\+ denotes that R is waiting for you to finish the previous command (not shown here)



Console Panes

Variables

- Save the results of a command in memory by **giving it a name**:

`z` uses `x` and `y`:

Values are not linked; updating `x` doesn't change `z`

Redefine `z` using the new value for `x`

Use " " or ' ' to distinguish between variable names and regular text

Note:
`x <- 3`: same as `x=3`

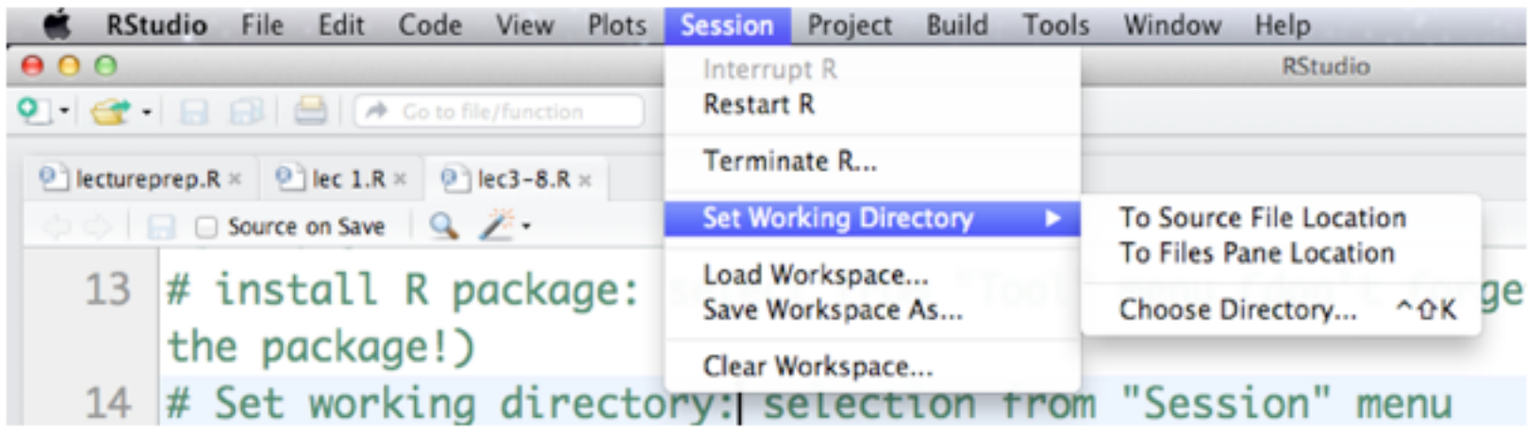
```
Console ~/
> x = 3
> y = 2
> x
[1] 3
> y
[1] 2
> z = sqrt(x^2+y^2)
> z
[1] 3.605551
> x = 5
> z
[1] 3.605551
> z = sqrt(x^2+y^2)
> z
[1] 5.385165
> str = 'Hi there'
> str
[1] "Hi there"
> str2 = "It's cold outside"
> str2
[1] "It's cold outside"
```

Create `x` and `y`

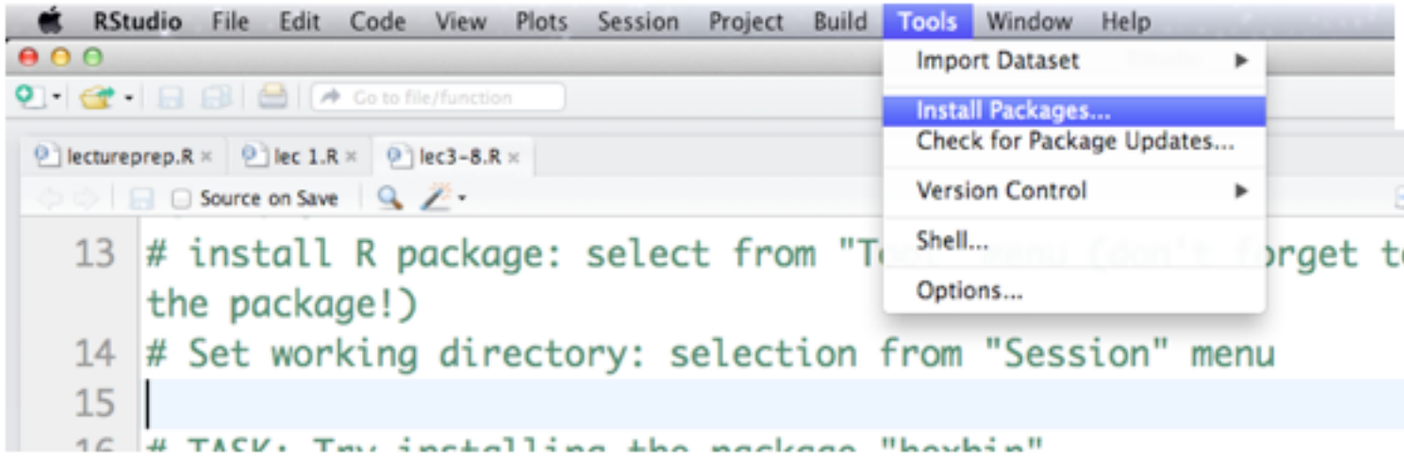
RStudio Menu

Two helpful menu items in Rstudio

- Set the current directory:



- Install a package



Packages are extensions to R, containing **new commands** for analysis or graphics

R Markdown

example.Rmd

Knit HTML | Chunks

```
Header 1
----------------------------------------------
This is an R Markdown document. Markdown is a
simple formatting syntax for authoring web pages.

Use an asterisk mark, to provide emphasis such as
*italics* and **bold**.

Create lists with a dash:
- Item 1
- Item 2
- Item 3

You can write `in-line` code with a back-tick.

```
Code blocks display
with fixed-width font
```

> Blockquotes are offset
```

RStudio: Preview HTML

Preview: ~/example.html

Header 1

This is an R Markdown document. Markdown is a simple formatting syntax for authoring web pages.

Use an asterisk mark, to provide emphasis such as *italics* and **bold**.

Create lists with a dash:

- Item 1
- Item 2
- Item 3

You can write `in-line` code with a back-tick.

```
Code blocks display
with fixed-width font
```

> Blockquotes are offset

R Markdown

chunks.Rmd

Knit HTML | Chunks

```
R Code Chunks
==========

With R Markdown, you can insert R code chunks including plots:

```{r qplot, fig.width=4, fig.height=3, message=FALSE}
# quick summary and plot
library(ggplot2)
summary(cars)
qplot(speed, dist, data=cars) +
      geom_smooth()
```
```

RStudio: Preview HTML

Preview: ~/chunks.html | Save As | Publish

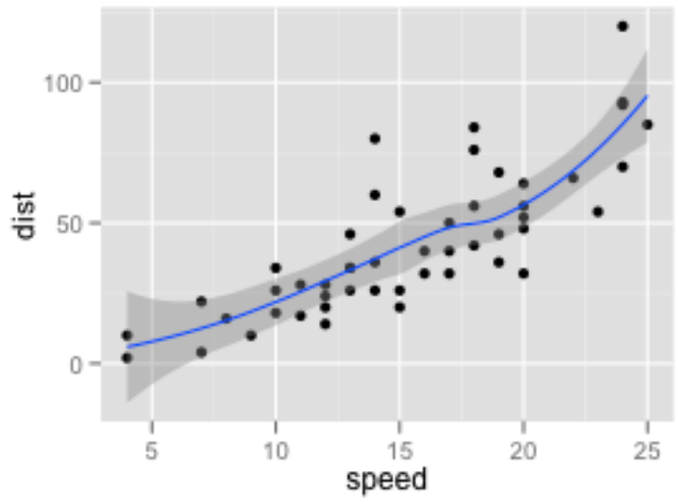
R Code Chunks

With R Markdown, you can insert R code chunks including plots:

```
# quick summary and plot
library(ggplot2)
summary(cars)
```

```
##      speed           dist
##  Min.   : 4.0   Min.   :  2
##  1st Qu.:12.0   1st Qu.: 26
##  Median :15.0   Median : 36
##  Mean   :15.4   Mean   : 43
##  3rd Qu.:19.0   3rd Qu.: 56
##  Max.   :25.0   Max.   :120
```

```
qplot(speed, dist, data = cars) + geom_smooth()
```



Workshop 2: Hello World

Open RStudio on your machine

- File > New File > R Markdown ...

- Change **summary(cars)** in the first code block to **print("Hello world!")**

- Click Knit HTML to produce an HTML file.

- Save your Rmd file as helloworld.Rmd

Data Types

You'll encounter different kinds of data types

- **Booleans** Direct binary values: TRUE or FALSE in R
- **Integers**: whole numbers (positive, negative or zero)
- **Characters** fixed-length blocks of bits, with special coding; strings = sequences of characters
- **Floating point numbers**: a fraction (with a finite number of bits) times an exponent, like 1.87 * 10^6
- **Missing** or ill-defined values: NA, NaN, etc.

Operators (Functions)

Command	Description
`+,-,*, /`	add, subtract, multiply, divide
`^`	raise to the power of
`%%`	remainder after division (ex: `8 %% 3 = 2`)
`( )`	change the order of operations
`log(), exp()`	logarithms and exponents (ex: `log(10) = 2.302`)
`sqrt()`	square root
`round()`	round to the nearest whole number (ex: `round(2.3) = 2`)
`floor(), ceiling()`	round down or round up
`abs()`	absolute value

Operators

```
7 + 5 # Addition
```

```
[1] 12
```

```
7 - 5 # Subtraction
```

```
[1] 2
```

```
7 * 5 # Multiplication
```

```
[1] 35
```

```
7 ^ 5 # Exponentiation
```

```
[1] 16807
```

```r
7 / 5 # Division
```

```
[1] 1.4
```

```r
7 %% 5 # Modulus
```

```
[1] 2
```

```r
7 %/% 5 # Integer division
```

```
[1] 1
```

Operators

Comparisons are also **binary operators**; they take two objects, like numbers. and give a Boolean

```
7 > 5
```

```
[1] TRUE
```

```
7 < 5
```

```
[1] FALSE
```

```
7 >= 7
```

```
[1] TRUE
```

```
7 <= 5
```

```
[1] FALSE
```

```
7 == 5
```

```
[1] FALSE
```

```
7 != 5
```

```
[1] TRUE
```

Boolean Operators

Basically “and” and “or”:

```
(5 > 7) & (6*7 == 42)
```

```
[1] FALSE
```

```
(5 > 7) | (6*7 == 42)
```

```
[1] TRUE
```

Basic Math / Basic Logic

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

?Arithmetic

- ! NOT
- & bitwise AND
- | bitwise OR
- && short circuit AND
- || short circuit OR
- == equality
- != NOT equality

?Logic

Also try ?Syntax, ?Comparison

More types

typeof() function returns the type

is.foo() functions return Booleans for whether the argument is of type foo

as.foo() (tries to) “cast” its argument to type foo — to translate it sensibly into a foo-type value

```
typeof(7)
```

```
[1] "double"
```

```
is.numeric(7)
```

```
[1] TRUE
```

```
is.na(7)
```

```
[1] FALSE
```

```
is.character(7)
```

```
[1] FALSE
```

```
is.character("7")
```

```
[1] TRUE
```

```
is.character("seven")
```

```
[1] TRUE
```

```
is.na("seven")
```

```
[1] FALSE
```

Variables

We can give names to data objects; these give us **variables**

A few variables are built in:

```
pi
```

```
[1] 3.141593
```

Variables can be arguments to functions or operators, just like constants:

```
pi*10
```

```
[1] 31.41593
```

```
cos(pi)
```

```
[1] -1
```

Assignment Operator

Most variables are created with the **assignment operator**, <- or =

```
approx.pi <- 22 / 7
approx.pi
```

```
[1] 3.142857
```

```
diameter.in.cubits = 10
approx.pi*diameter.in.cubits
```

```
[1] 31.42857
```

Variables

- Using names and variables makes code: easier to design, easier to debug, less prone to bugs, easier to improve, and easier for others to read
- Avoid “magic constants”; use named variables
- Use descriptive variable names
  - Good: `num.students <- 35`
  - Bad: `ns <- 35`

Workspace

What names have you defined values for?

```
ls()
```

```
[1] "approx.pi"                 "circumference.in.cubits"
[3] "diameter.in.cubits"
```

Getting rid of variables:

```
rm("circumference.in.cubits")
ls()
```

```
[1] "approx.pi"           "diameter.in.cubits"
```

Data Structure : Vector

- Group related data values into one object, a **data structure**
- A **vector** is a sequence of values, all of the same type
- `c()` function returns a vector containing all its arguments in order

```
students <- c("Sean", "Louisa", "Frank", "Farhad", "Li")
midterm <- c(80, 90, 93, 82, 95)
```

- Typing the variable name at the prompt causes it to display

```
students
```

```
[1] "Sean"   "Louisa" "Frank"  "Farhad" "Li"
```

Indexing Vector

- `vec[1]` is the first element, `vec[4]` is the 4th element of `vec`

```
students
```

```
[1] "Sean"   "Louisa" "Frank"  "Farhad" "Li"
```

```
students[4]
```

```
[1] "Farhad"
```

- `vec[-4]` is a vector containing all but the fourth element

```
students[-4]
```

```
[1] "Sean"   "Louisa" "Frank"  "Li"
```

Vector Arithmetic

Operators apply to vectors “pairwise” or “elementwise”:

```
final <- c(78, 84, 95, 82, 91) # Final exam scores
midterm # Midterm exam scores
```

```
[1] 80 90 93 82 95
```

```
midterm + final # Sum of midterm and final scores
```

```
[1] 158 174 188 164 186
```

```
(midterm + final)/2 # Average exam score
```

```
[1] 79 87 94 82 93
```

```
course.grades <- 0.4*midterm + 0.6*final # Final course grade
course.grades
```

```
[1] 78.8 86.4 94.2 82.0 92.6
```

Pairwise Comparison

Is the final score higher than the midterm score?

```
midterm
```

```
[1] 80 90 93 82 95
```

```
final
```

```
[1] 78 84 95 82 91
```

```
final > midterm
```

```
[1] FALSE FALSE  TRUE FALSE FALSE
```

Boolean operators can be applied elementwise:

```
(final < midterm) & (midterm > 80)
```

```
[1] FALSE  TRUE FALSE FALSE  TRUE
```

Functions on Vectors

Command	Description
`sum(vec)`	sums up all the elements of `vec`
`mean(vec)`	mean of `vec`
`median(vec)`	median of `vec`
`min(vec), max(vec)`	the largest or smallest element of `vec`
`sd(vec), var(vec)`	the standard deviation and variance of `vec`
`length(vec)`	the number of elements in `vec`
`pmax(vec1, vec2), pmin(vec1, vec2)`	example: `pmax(quiz1, quiz2)` returns the higher of quiz 1 and quiz 2 for each student
`sort(vec)`	returns the `vec` in sorted order
`order(vec)`	returns the index that sorts the vector `vec`
`unique(vec)`	lists the unique elements of `vec`
`summary(vec)`	gives a five-number summary
`any(vec), all(vec)`	useful on Boolean vectors

Function on Vectors

```
course.grades
```

```
[1] 78.8 86.4 94.2 82.0 92.6
```

```
mean(course.grades) # mean grade
```

```
[1] 86.8
```

```
median(course.grades)
```

```
[1] 86.4
```

```
sd(course.grades) # grade standard deviation
```

```
[1] 6.625708
```

More on Functions

```
sort(course.grades)
```

```
[1] 78.8 82.0 86.4 92.6 94.2
```

```
max(course.grades) # highest course grade
```

```
[1] 94.2
```

```
min(course.grades) # lowest course grade
```

```
[1] 78.8
```

Referencing elements of Vectors

```
students
```

```
[1] "Sean"   "Louisa" "Frank"  "Farhad" "Li"
```

Vector of indices:

```
students[c(2,4)]
```

```
[1] "Louisa" "Farhad"
```

Vector of negative indices

```
students[c(-1,-3)]
```

```
[1] "Louisa" "Farhad" "Li"
```

More on Referencing

`which()` returns the `TRUE` indexes of a Boolean vector:

```
course.grades
```

```
[1] 78.8 86.4 94.2 82.0 92.6
```

```
a.threshold <- 90 # A grade = 90% or higher
course.grades >= a.threshold # vector of booleans
```

```
[1] FALSE FALSE  TRUE FALSE  TRUE
```

```
a.students <- which(course.grades >= a.threshold) # Applying which()
a.students
```

```
[1] 3 5
```

```
students[a.students] # Names of A students
```

```
[1] "Frank" "Li"
```

Workshop 3

Using the same Rmarkdown file from Workshop 1, do the following:-

1) Creating sequences

: Colon operator:

```
1:10 # Numbers 1 to 10
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```
127:132 # Numbers 127 to 132
```

```
## [1] 127 128 129 130 131 132
```

Workshop 3

`seq` function: `seq(from, to, by)`

```
seq(1,10,1) # Numbers 1 to 10
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```
seq(1,10,2) # Odd numbers from 1 to 10
```

```
## [1] 1 3 5 7 9
```

```
seq(2,10,2) # Even numbers from 2 to 10
```

```
## [1]  2  4  6  8 10
```

Workshop 3 : Use RStudio to answer questions below

(a) Use : to output the sequence of numbers from 3 to 12

(b) Use seq() to output the sequence of numbers from 3 to 30 in increments of 3

(c) Save the sequence from (a) as a variable x, and the sequence from (b) as a variable y. Output their product x*y

Difference between = and <-

- The operators <- and = assign into the environment in which they are evaluated.
- The operator <- can be used anywhere, whereas the operator = is only allowed at the top level (e.g., in the complete expression typed at the command prompt) or as one of the subexpressions in a braced list of expressions.

```
matrix(1,nrow=2)

matrix(1,nrow<-2)
```

Reading and Getting Data into R

Combine Command

c(1, 2, 3, 4)
c(item1, item2, item3, item4)
c("item1", "item2", "item3")

Scan Command

our.data = scan()

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

Working Directory

getwd()

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')
```

Importing Data

- To import tabular data into R, we use the **`read.table()`** command

```
1
2 survey <- read.table("survey_data.csv", header=TRUE, sep=",")
3
```

- Let's parse this command one component at a time
  - The data is in a file called `survey_data.csv`, which is an online file
  - The file contains a `header` as its first row
  - The csv format means that the data is comma-separated, so `sep=","`
- Could've also used `read.csv()`, which is just `read.table()` with the preset `sep=","`

Exploring the Data

- R imports data into a **`data.frame`** object

```
class(survey)
```

```
[1] "data.frame"
```

- To view the first few rows of the data, use **`head()`**

```
head(survey, 3)
```

```
  Program              PriorExp      Rexperience OperatingSystem TVhours
1    MISM       Some experience       Never used          Windows       1
2   Other       Some experience Basic competence          Windows       8
3    MISM Extensive experience Basic competence          Windows       4
          Editor
1 Microsoft Word
2 Microsoft Word
3 Microsoft Word
```

- `head(data.frame, n)` returns the first `n` rows of the data frame

- In the Console, you can also use `View(survey)` to get a spreadsheet view

Simple Summary

- Use the `str()` function to get a simple summary of your data set

```
str(survey)
```

```
'data.frame':   31 obs. of  6 variables:
 $ Program        : Factor w/ 3 levels "MISM","Other",..: 1 2 1 3 2 3 2 1 3 3 ...
 $ PriorExp       : Factor w/ 3 levels "Extensive experience",..: 3 3 1 2 2 3 3 3
2 1 ...
 $ Rexperience    : Factor w/ 3 levels "Basic competence",..: 3 1 1 2 3 3 3 3 3 3
...
 $ OperatingSystem: Factor w/ 2 levels "Mac OS X","Windows": 2 2 2 2 2 1 1 2 1 2
...
 $ TVhours        : int  1 8 4 5 3 10 0 10 15 4 ...
 $ Editor         : Factor w/ 1 level "Microsoft Word": 1 1 1 1 1 1 1 1 1 1 ...
```

- This says that TVhours is a numeric variable, while all the rest are factors (categorical)

Another Summary

```
summary(survey)
```

```
 Program                        PriorExp                    Rexperience
MISM :10   Extensive experience    : 5   Basic competence     : 2
Other: 6   Never programmed before:12   Installed on machine: 7
PPM  :15   Some experience         :14   Never used           :22


OperatingSystem    TVhours                     Editor
Mac OS X:14     Min.    : 0.000   Microsoft Word:31
Windows :17     1st Qu.: 1.000
                Median : 4.000
                Mean   : 5.742
                3rd Qu.: 9.000
                Max.   :40.000
```

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.
  - To see what an R object is made up of, you can use `attributes()`

```
attributes(survey)
```

```
$names
[1] "Program"          "PriorExp"         "Rexperience"      "OperatingSystem"
[5] "TVhours"          "Editor"

$class
[1] "data.frame"

$row.names
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
[24] 24 25 26 27 28 29 30 31
```

An R **data frame** is a *list* whose columns you can refer to by *name* or *index*

Data Frame dimensions

- We can use `nrow()` and `ncol` to determine the number of survey responses and the number of survey questions

```
nrow(survey) # Number of rows (responses)
```

```
[1] 31
```

```
ncol(survey) # Number of columns (questions)
```

```
[1] 6
```

- When writing reports, you will often want to say how large your sample size was
- To do this *inline*, use the syntax:

```
`r nrow(survey)`
```

- This allows us to write "31 students responded to the survey", and have the number displayed automatically change when `nrow(survey)` changes.

Indexing Data Frame

- There are many different ways of indexing the same piece of a data frame

```
survey[["Program"]]  # "Program" element
```

```
 [1] MISM  Other MISM  PPM   Other PPM   Other MISM  PPM   PPM   MISM
[12] PPM   PPM   PPM   PPM   PPM   PPM   MISM  MISM  MISM  PPM   PPM
[23] MISM  Other PPM   PPM   MISM  PPM   Other Other MISM
Levels: MISM Other PPM
```

```
survey$Program # "Program" element
```

```
 [1] MISM  Other MISM  PPM   Other PPM   Other MISM  PPM   PPM   MISM
[12] PPM   PPM   PPM   PPM   PPM   PPM   MISM  MISM  MISM  PPM   PPM
[23] MISM  Other PPM   PPM   MISM  PPM   Other Other MISM
Levels: MISM Other PPM
```

```
survey[,1] # Data from 1st column
```

```
 [1] MISM  Other MISM  PPM   Other PPM   Other MISM  PPM   PPM   MISM
[12] PPM   PPM   PPM   PPM   PPM   PPM   MISM  MISM  MISM  PPM   PPM
[23] MISM  Other PPM   PPM   MISM  PPM   Other Other MISM
Levels: MISM Other PPM
```

More Indexing

- Note that single brackets and double brackets have different effects

```
survey[["Program"]]
```

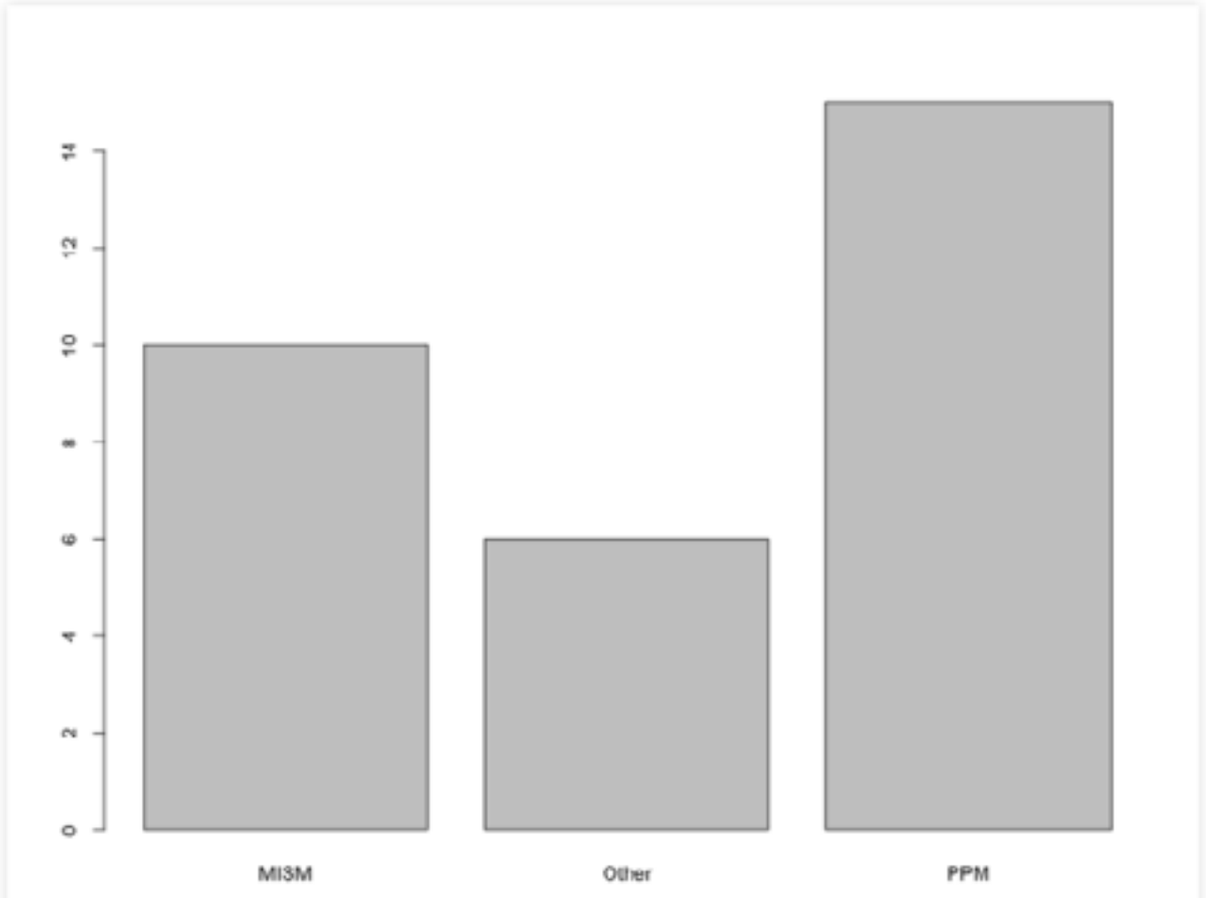
```
 [1] MISM  Other MISM  PPM   Other PPM   Other MISM  PPM   PPM   MISM
[12] PPM   PPM   PPM   PPM   PPM   PPM   MISM  MISM  MISM  PPM   PPM
[23] MISM  Other PPM   PPM   MISM  PPM   Other Other MISM
Levels: MISM Other PPM
```

```
survey["Program"] # sub-data frame containing only "Program"
```

```
   Program
1     MISM
2    Other
3     MISM
4      PPM
5    Other
6      PPM
7    Other
8     MISM
9      PPM
10     PPM
11    MISM
12     PPM
13     PPM
14     PPM
```

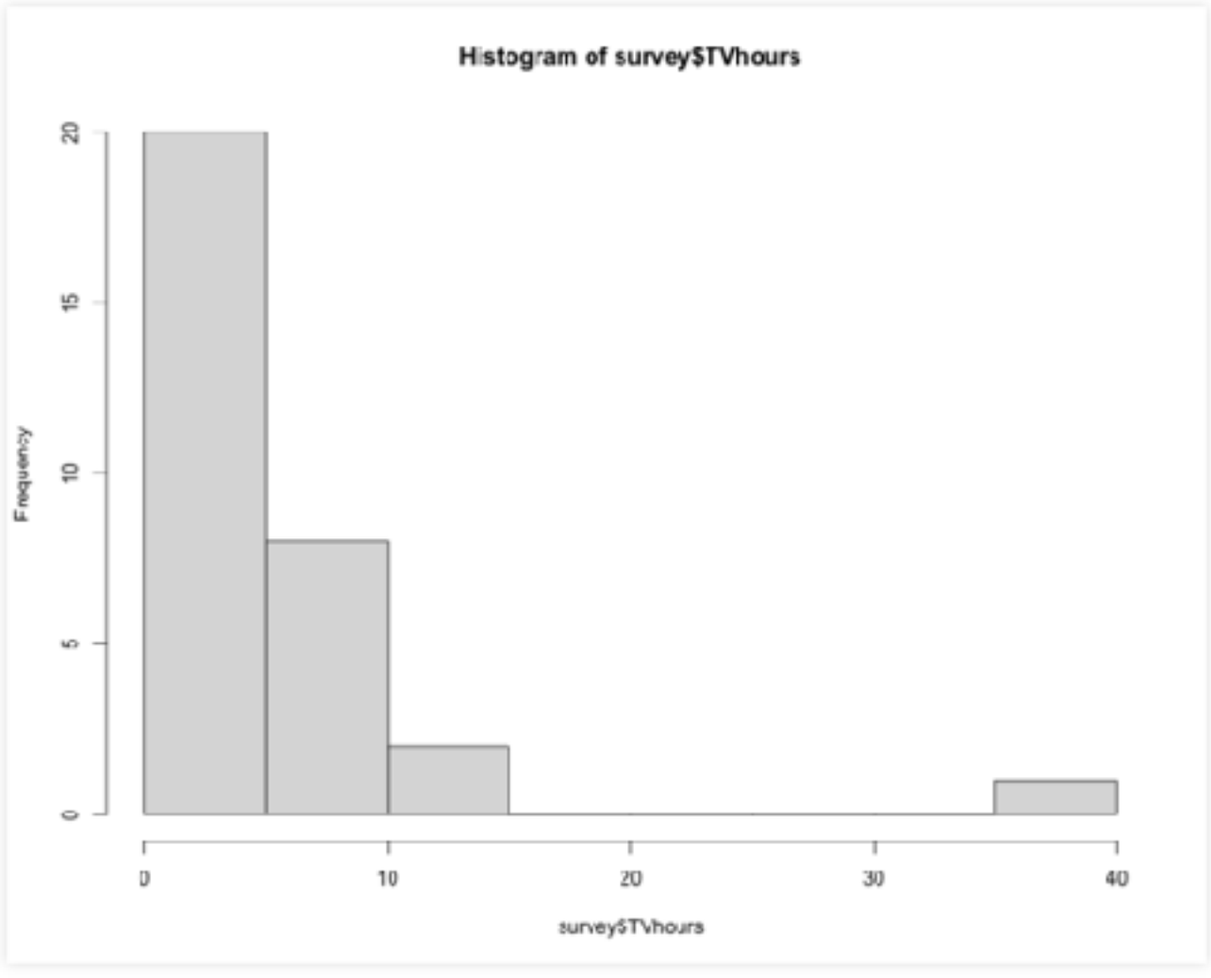
Bar Plot (Categorical Data)

```
plot(survey[["Program"]])
```



Histogram (Continuous Data)

```
hist(survey$TVhours, col="lightgray")
```



Indexing Multiple Columns

```
head(survey[,c(1,5)]) # Data from 1st and 5th columns
```

```
  Program TVhours
1    MISM        1
2   Other        8
3    MISM        4
4     PPM        5
5   Other        3
6     PPM       10
```

```
head(survey[c("Program", "Editor")]) # Data from "Program" and "Editor"
```

```
  Program          Editor
1    MISM Microsoft Word
2   Other Microsoft Word
3    MISM Microsoft Word
4     PPM Microsoft Word
5   Other Microsoft Word
6     PPM Microsoft Word
```

Indexing Row and Column

- Data frames have two dimensions to index across

```
survey[6,] # 6th row
```

```
  Program        PriorExp Rexperience OperatingSystem TVhours
6     PPM Some experience  Never used         Mac OS X       10
          Editor
6 Microsoft Word
```

```
survey[6,5] # row 6, column 5
```

```
[1] 10
```

```
survey[6, "Program"] # Program of 6th survey respondent
```

```
[1] PPM
Levels: MISM Other PPM
```

```
survey[["Program"]][6]  # Program of 6th survey respondent
```

```
[1] PPM
Levels: MISM Other PPM
```

More Indexing

- We can use this operator for indexing

```
survey[1:3,]  # equivalent to head(survey, 3)
```

```
  Program              PriorExp      Rexperience OperatingSystem TVhours
1    MISM       Some experience       Never used          Windows        1
2   Other       Some experience Basic competence          Windows        8
3    MISM Extensive experience Basic competence          Windows        4
          Editor
1 Microsoft Word
2 Microsoft Word
3 Microsoft Word
```

```
survey[3:5, c(1,5)]
```

```
  Program TVhours
3    MISM        4
4     PPM        5
5   Other        3
```

Subsets of Data

- We are often interested in learning something a specific subset of the data

```
survey[survey$Program=="MISM",] # Data from the MISM students
```

```
   Program              PriorExp          Rexperience OperatingSystem
1     MISM      Some experience           Never used         Windows
3     MISM Extensive experience    Basic competence         Windows
8     MISM      Some experience           Never used         Windows
11    MISM Extensive experience Installed on machine       Mac OS X
18    MISM Extensive experience           Never used       Mac OS X
19    MISM Extensive experience           Never used         Windows
20    MISM      Some experience           Never used         Windows
23    MISM      Some experience           Never used         Windows
27    MISM      Some experience           Never used       Mac OS X
31    MISM      Some experience Installed on machine         Windows
   TVhours          Editor
1         1 Microsoft Word
3         4 Microsoft Word
8        10 Microsoft Word
11        0 Microsoft Word
18        3 Microsoft Word
19        0 Microsoft Word
20        1 Microsoft Word
23        3 Microsoft Word
27        0 Microsoft Word
31        0 Microsoft Word
```

More subsets example

- Let's pull all of the PPM students who have never used R before

```
survey[survey$Program=="PPM" & survey$Rexperience=="Never used",]
```

```
   Program                PriorExp Rexperience OperatingSystem TVhours
6      PPM          Some experience  Never used         Mac OS X       10
9      PPM Never programmed before  Never used         Mac OS X       15
10     PPM    Extensive experience  Never used          Windows        4
12     PPM Never programmed before  Never used          Windows        0
13     PPM          Some experience  Never used         Mac OS X       10
14     PPM Never programmed before  Never used         Mac OS X        4
15     PPM          Some experience  Never used          Windows       10
16     PPM          Some experience  Never used         Mac OS X        2
22     PPM Never programmed before  Never used          Windows        7
25     PPM Never programmed before  Never used         Mac OS X        6
           Editor
6  Microsoft Word
9  Microsoft Word
10 Microsoft Word
12 Microsoft Word
13 Microsoft Word
14 Microsoft Word
15 Microsoft Word
16 Microsoft Word
22 Microsoft Word
25 Microsoft Word
```

Subset

- When the subset conditions get long or messy, it is preferable to use the `subset()` function

- Here's an example of selecting the OperatingSystem and TVhours responses from all of the students who are either in PPM or Other and who listed their R experience as “Basic competence”.

```
subset(survey, select=c("OperatingSystem", "TVhours"),  subset=(Program == "PPM"
| Program == "Other") & Rexperience == "Basic competence")
```

```
  OperatingSystem TVhours
2         Windows       8
```

Calculation from subset

```
mean(survey$TVhours[survey$Program == "PPM"]) # Average time PPM's spent watching
TV
```

```
[1] 8.8
```

```
mean(survey$TVhours[survey$Program == "MISM"]) # Average time MISM's spent
watching TV
```

```
[1] 2.2
```

```
mean(survey$TVhours[survey$Program == "Other"]) # Average time "Others" spent
watching TV
```

```
[1] 4
```

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)

Example Style Guide

Assignment operator. USE <-

```
student.names <- c("Eric", "Hao", "Jennifer")  # Good
student.names = c("Eric", "Hao", "Jennifer") # Bad
```

- Note: When specifying function arguments, only = is valid

```
sort(tv.hours, decreasing=TRUE) # Good
sort(tv.hours, decreasing<-TRUE) # Bad!!
```

Style Guide : Variable Name

- To make code easy to read, debug, and maintain, you should use **concise** but **descriptive** variable names

- Terms in variable names should be separated by _ or .

```
# Accepted
day_one    day.one    day_1    day.1    day1

# Bad
d1    DayOne    dayone

# Can be made more concise:
first.day.of.the.month
```

- Avoid using variable names that are already pre-defined in R

```
# EXTREMELY bad:
c    T    pi    sum    mean
```

Workshop 4: Importing and Indexing data

Data practice

survey <- read.csv("survey_data.csv", header=TRUE);

Use command to answer these questions?

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

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

Index practice

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

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

More on Data Frame

```
library(MASS)
head(Cars93, 3)
```

```
  Manufacturer   Model     Type Min.Price Price Max.Price MPG.city
1        Acura Integra    Small       12.9  15.9       18.8       25
2        Acura  Legend Midsize       29.2  33.9       38.7       18
3         Audi      90 Compact       25.9  29.1       32.3       20
  MPG.highway             AirBags DriveTrain Cylinders EngineSize
1          31                None      Front         4        1.8
2          25 Driver & Passenger      Front         6        3.2
3          26         Driver only      Front         6        2.8
  Horsepower  RPM Rev.per.mile Man.trans.avail Fuel.tank.capacity
1        140 6300         2890             Yes                13.2
2        200 5500         2335             Yes                18.0
3        172 5500         2280             Yes                16.9
  Passengers Length Wheelbase Width Turn.circle Rear.seat.room
1          5    177       102    68           37           26.5
2          5    195       115    71           38           30.0
3          5    180       102    67           37           28.0
  Luggage.room Weight  Origin           Make
1           11   2705 non-USA Acura Integra
2           15   3560 non-USA  Acura Legend
3           14   3375 non-USA       Audi 90
```

Add column using "Transform"

- `transform()` returns a new data frame with columns modified or added as specified by the function call

```
Cars93.metric <- transform(Cars93,
                           KMPL.city = 0.425 * MPG.city,
                           KMPL.highway = 0.425 * MPG.highway)
tail(names(Cars93.metric))
```

```
[1] "Luggage.room" "Weight"       "Origin"       "Make"
[5] "KMPL.city"    "KMPL.highway"
```

- Our data frame has two new columns, giving the fuel consumption in km/l

Another Approach

```
KMPL.city.2 <- 0.425 * Cars93$MPG.city
# Add a new column called KMPL.city.2
Cars93.metric$KMPL.city.2 <- KMPL.city.2
tail(names(Cars93.metric))
```

```
[1] "Weight"       "Origin"       "Make"         "KMPL.city"
[5] "KMPL.highway" "KMPL.city.2"
```

- Let's check that both approaches did the same thing

```
identical(Cars93.metric$KMPL.city, Cars93.metric$KMPL.city.2)
```

```
[1] TRUE
```

Changing level of a factor

```
manufacturer <- Cars93$Manufacturer
head(manufacturer, 10)
```

```
 [1] Acura     Acura     Audi      Audi      BMW       Buick     Buick
 [8] Buick     Buick     Cadillac
32 Levels: Acura Audi BMW Buick Cadillac Chevrolet Chrylser ... Volvo
```

We'll use the `mapvalues(x, from, to)` function from the `plyr` library.

```
library(plyr)

# Map Chevrolet, Pontiac and Buick to GM
manufacturer.combined <- mapvalues(manufacturer,
                                   from = c("Chevrolet", "Pontiac", "Buick"),
                                   to = rep("GM", 3))

head(manufacturer.combined, 10)
```

```
 [1] Acura     Acura     Audi      Audi      BMW       GM        GM
 [8] GM        GM        Cadillac
30 Levels: Acura Audi BMW GM Cadillac Chrylser Chrysler Dodge ... Volvo
```

Another example

- A lot of data comes with integer encodings of levels
- You may want to convert the integers to more meaningful values for the purpose of your analysis
- Let's pretend that in the class survey 'Program' was coded as an integer with 1 = MISM, 2 = Other, 3 = PPM
- Here's how we would get back the program codings using the transform(), as.factor() and mapvalues() functions

```
survey <- transform(survey, Program = as.factor(mapvalues(Program, c(1, 2, 3),
c("MISM", "Other", "PPM"))))
head(survey)
```

```
  Program                 PriorExp          Rexperience OperatingSystem
1    MISM          Some experience           Never used          Windows
2   Other          Some experience      Basic competence          Windows
3    MISM    Extensive experience      Basic competence          Windows
4     PPM Never programmed before Installed on machine          Windows
5   Other Never programmed before           Never used          Windows
6     PPM          Some experience           Never used        Mac OS X
  TVhours          Editor
1       1 Microsoft Word
2       8 Microsoft Word
3       4 Microsoft Word
4       5 Microsoft Word
5       3 Microsoft Word
6      10 Microsoft Word
```

table() function

- The `table()` function builds **contingency tables** showing counts at each combination of factor levels

```
table(Cars93$AirBags)
```

```
Driver & Passenger        Driver only               None
                16                 43                 34
```

```
table(Cars93$Origin)
```

```
    USA non-USA
     48      45
```

```
table(Cars93$AirBags, Cars93$Origin)
```

```
                    USA non-USA
Driver & Passenger    9       7
Driver only          23      20
None                 16      18
```

- Looks like US and non-US cars had about the same distribution of AirBag types

more table()

- When `table()` is supplied a data frame, it produces contingency tables for all combinations of factors

```
head(Cars93[c("AirBags", "Origin")], 3)
```

```
              AirBags  Origin
1                None non-USA
2 Driver & Passenger non-USA
3         Driver only non-USA
```

```
table(Cars93[c("AirBags", "Origin")])
```

```
                    Origin
AirBags              USA non-USA
  Driver & Passenger   9       7
  Driver only         23      20
  None                16      18
```

Basic of Lists

A list is a **data structure** that can be used to store **different kinds** of data

- Recall: a vector is a data structure for storing *similar kinds of data*
- To better understand the difference, consider the following example.

```
my.vector.1 <- c("Michael", 165, TRUE) # (name, weight, is.male)
my.vector.1
```

```
[1] "Michael" "165"     "TRUE"
```

```
typeof(my.vector.1)  # All the elements are now character strings!
```

```
[1] "character"
```

List vs Vector

```
my.vector.2 <- c(FALSE, TRUE, 27) # (is.male, is.citizen, age)
typeof(my.vector.2)
```

```
[1] "double"
```

- Vectors expect elements to be all of the same type (e.g., `Boolean`, `numeric`, `character`)
- When data of different types are put into a vector, the R converts everything to a common type

Lists

- To store data of different types in the same object, we use lists
- Simple way to build lists: use `list()` function

```
my.list <- list("Michael", 165, TRUE)
my.list
```

```
[[1]]
[1] "Michael"

[[2]]
[1] 165

[[3]]
[1] TRUE
```

```
sapply(my.list, typeof)
```

```
[1] "character" "double"    "logical"
```

Named Elements

```
patient.1 <- list(name="Michael", weight=165, is.male=TRUE)
patient.1
```

```
$name
[1] "Michael"

$weight
[1] 165

$is.male
[1] TRUE
```

Referencing element inside list

```
patient.1$name # Get "name" element (returns a string)
```

```
[1] "Michael"
```

```
patient.1[["name"]] # Get "name" element (returns a string)
```

```
[1] "Michael"
```

```
patient.1["name"] # Get "name" slice (returns a sub-list)
```

```
$name
[1] "Michael"
```

```
c(typeof(patient.1$name), typeof(patient.1["name"]))
```

```
[1] "character" "list"
```

Function

- We have used a lot of built-in functions: `mean()`, `subset()`, `plot()`, `read.table()`...
- An important part of programming and data analysis is to write custom functions
- Functions help make code **modular**
- Functions make debugging easier
- Remember: this entire class is about applying *functions* to *data*

what is a function?

A function is a machine that turns **input objects** (arguments) into an **output object** (return value) according to a definite rule.

- Let's look at a really simple function

```
addOne <- function(x) {
  x + 1
}
```

- `x` is the **argument** or **input**
- The function **output** is the input `x` incremented by 1

```
addOne(12)
```

```
[1] 13
```

Another example

- Here's a function that returns a % given a numerator, denominator, and desired number of decimal values

```
calculatePercentage <- function(x, y, d) {
  decimal <- x / y  # Calculate decimal value
  round(100 * decimal, d)  # Convert to % and round to d digits
}
```

```
calculatePercentage(27, 80, 1)
```

```
[1] 33.8
```

- If you're calculating several %'s for your report, you should use this kind of function instead of repeatedly copying and pasting code

Function return a list

- Here's a function that takes a person's full name (FirstName LastName), weight in lb and height in inches and converts it into a list with the person's first name, person's last name, weight in kg, height in m, and BMI.

```
createPatientRecord <- function(full.name, weight, height) {
  name.list <- strsplit(full.name, split=" ")[[1]]
  first.name <- name.list[1]
  last.name <- name.list[2]
  weight.in.kg <- weight / 2.2
  height.in.m <- height * 0.0254
  bmi <- weight.in.kg / (height.in.m ^ 2)
  list(first.name=first.name, last.name=last.name, weight=weight.in.kg,
height=height.in.m,
       bmi=bmi)
}
```

Try out function

```
createPatientRecord("Michael Smith", 185, 12 * 6 + 1)
```

```
$first.name
[1] "Michael"

$last.name
[1] "Smith"

$weight
[1] 84.09091

$height
[1] 1.8542

$bmi
[1] 24.45884
```

Another example

- Calculate mean, median and standard deviation

```
threeNumberSummary <- function(x) {
  c(mean=mean(x), median=median(x), sd=sd(x))
}
x <- rnorm(100, mean=5, sd=2) # Vector of 100 normals with mean 5 and sd 2
threeNumberSummary(x)
```

```
    mean   median       sd
4.926875 5.050984 1.953703
```

If-else statement

- Oftentimes we want our code to have different effects depending on the features of the input
- Example: Calculating a student's letter grade
  - If grade >= 90, assign A
  - Otherwise, if grade >= 80, assign B
  - Otherwise, if grade >= 70, assign C
  - In all other cases, assign F
- To code this up, we use if-else statements

If-else example

```
calculateLetterGrade <- function(x) {
  if(x >= 90) {
    grade <- "A"
  } else if(x >= 80) {
    grade <- "B"
  } else if(x >= 70) {
    grade <- "C"
  } else {
    grade <- "F"
  }
  grade
}

course.grades <- c(92, 78, 87, 91, 62)
sapply(course.grades, FUN=calculateLetterGrade)
```

```
[1] "A" "C" "B" "A" "F"
```

return()

- In the previous examples we specified the output simply by writing the output variable as the last line of the function
- More explicitly, we can use the `return()` function

```
addOne <- function(x) {
  return(x + 1)
}

addOne(12)
```

```
[1] 13
```

- We will generally avoid the `return()` function, but you can use it if necessary or if it makes writing a particular function easier.

Workshop 5 : Tranform

For the first two problems we'll use the Cars93 data set from the MASS library.

```
library(MASS)
```

1. **Manipulating data frames**

Use the **transform() and log()** functions to create a new data frame called Cars93.log that has MPG.highway and MPG.city replaced with log(MPG.highway) and log(MPG.city).

2. **Functions, lists, and if-else**

(a) Write a function called **isPassingGrade** whose input x is a number, and which returns FALSE if x is lower than 50 and TRUE otherwise.

(b) Write a function called **sendMessage** whose input x is a number, and which prints **Congratulations** if **isPassingGrade(x)** is TRUE and prints **Oh no!** if **isPassingGrade(x)** is FALSE.

Data Wrangling

Common Problem

- One of the most common problems you'll encounter when importing manually-entered data is inconsistent data types within columns
- For a simple example, let's look at TVhours column in a messy version of the survey data

```
survey.messy <- read.csv("survey_messy.csv", header=TRUE)
survey.messy$TVhours
```

```
 [1] 1                  8h                 4                  5
 [5] 3                  ~10                0                  10
 [9] 15 (incl movies)   4                  0                  0
[13] 10                 4                  10                 2hours
[17] 5                  3                  0                  1
[21] 2                  7                  3                  10
[25] 6.5                40                 0                  12
[29] adfjalkj           3                  0
16 Levels: ~10 0 1 10 12 15 (incl movies) 2 2hours 3 4 40 5 6.5 7 ... adfjalkj
```

What are the problems?

```
str(survey.messy)
```

```
'data.frame':   31 obs. of  6 variables:
 $ Program         : Factor w/ 3 levels "MISM","Other",..: 1 2 1 3 2 3 2 1 3 3 ...
 $ PriorExp        : Factor w/ 3 levels "Extensive experience",..: 3 3 1 2 2 3 3 3
2 1 ...
 $ Rexperience     : Factor w/ 3 levels "Basic competence",..: 3 1 1 2 3 3 3 3 3 3
...
 $ OperatingSystem: Factor w/ 2 levels "Mac OS X","Windows": 2 2 2 2 2 1 1 2 1 2
...
 $ TVhours         : Factor w/ 16 levels "~10","0","1",..: 3 15 10 12 9 1 2 4 6 10
...
 $ Editor          : Factor w/ 1 level "Microsoft Word": 1 1 1 1 1 1 1 1 1 1 ...
```

- Several of the entries have non-numeric values in them (they contain strings)
- As a result, `TVhours` is being imported as `factor`

Fix the type

```
as.character(tv.hours.messy)[1:30]
```

```
 [1] "1"                  "8h"                 "4"
 [4] "5"                  "3"                  "~10"
 [7] "0"                  "10"                 "15 (incl movies)"
[10] "4"                  "0"                  "0"
[13] "10"                 "4"                  "10"
[16] "2hours"             "5"                  "3"
[19] "0"                  "1"                  "2"
[22] "7"                  "3"                  "10"
[25] "6.5"                "40"                 "0"
[28] "12"                 "adfjalkj"           "3"
```

```
as.numeric(as.character(tv.hours.messy))[1:30]
```

```
 [1]  1.0   NA  4.0  5.0  3.0   NA  0.0 10.0   NA  4.0  0.0  0.0 10.0  4.0
[15] 10.0   NA  5.0  3.0  0.0  1.0  2.0  7.0  3.0 10.0  6.5 40.0  0.0 12.0
[29]   NA  3.0
```

```
typeof(as.numeric(as.character(tv.hours.messy)))  # Success!! (Almost...)
```

```
[1] "double"
```

A small improvement

- All the corrupted cells now appear as `NA`, which is R's missing indicator
- We can do a little better by cleaning up the vector once we get it to `character` form

```
tv.hours.strings <- as.character(tv.hours.messy)
tv.hours.strings
```

```
 [1] "1"          "8h"          "4"
 [4] "5"          "3"           "~10"
 [7] "0"          "10"          "15 (incl movies)"
[10] "4"          "0"           "0"
[13] "10"         "4"           "10"
[16] "2hours"     "5"           "3"
[19] "0"          "1"           "2"
[22] "7"          "3"           "10"
[25] "6.5"        "40"          "0"
[28] "12"         "adfjalkj"    "3"
[31] "0"
```

Deleting non-numeric (or .) characters

```
tv.hours.strings
```

```
 [1] "1"          "8h"          "4"
 [4] "5"          "3"           "~10"
 [7] "0"          "10"          "15 (incl movies)"
[10] "4"          "0"           "0"
[13] "10"         "4"           "10"
[16] "2hours"     "5"           "3"
[19] "0"          "1"           "2"
[22] "7"          "3"           "10"
[25] "6.5"        "40"          "0"
[28] "12"         "adfjalkj"    "3"
[31] "0"
```

```
# Use gsub() to replace everything except digits and '.' with a blank ""
gsub("[^0-9.]", "", tv.hours.strings)
```

```
 [1] "1"   "8"   "4"   "5"   "3"   "10"  "0"   "10"  "15"  "4"   "0"
[12] "0"   "10"  "4"   "10"  "2"   "5"   "3"   "0"   "1"   "2"   "7"
[23] "3"   "10"  "6.5" "40"  "0"   "12"  ""    "3"   "0"
```

Redo

```
tv.hours.messy
```

```
 [1] 1                  8h                 4                  5
 [5] 3                  ~10                0                  10
 [9] 15 (incl movies)   4                  0                  0
[13] 10                 4                  10                 2hours
[17] 5                  3                  0                  1
[21] 2                  7                  3                  10
[25] 6.5                40                 0                  12
[29] adfjalkj           3                  0
16 Levels: ~10 0 1 10 12 15 (incl movies) 2 2hours 3 4 40 5 6.5 7 ... adfjalkj
```

```
tv.hours.clean <- as.numeric(gsub("[^0-9.]", "", tv.hours.strings))
tv.hours.clean
```

```
 [1]  1.0  8.0  4.0  5.0  3.0 10.0  0.0 10.0 15.0  4.0  0.0  0.0 10.0  4.0
[15] 10.0  2.0  5.0  3.0  0.0  1.0  2.0  7.0  3.0 10.0  6.5 40.0  0.0 12.0
[29]   NA  3.0  0.0
```

Another approach

- We can also handle this problem by setting `stringsAsFactors = FALSE` when importing our data.

```
survey.messy <- read.csv("survey_messy.csv", header=TRUE, stringsAsFactors=FALSE)
str(survey.messy)
```

```
'data.frame':   31 obs. of  6 variables:
 $ Program        : chr  "MISM" "Other" "MISM" "PPM" ...
 $ PriorExp       : chr  "Some experience" "Some experience" "Extensive
experience" "Never programmed before" ...
 $ Rexperience    : chr  "Never used" "Basic competence" "Basic competence"
"Installed on machine" ...
 $ OperatingSystem: chr  "Windows" "Windows" "Windows" "Windows" ...
 $ TVhours        : chr  "1" "8h" "4" "5" ...
 $ Editor         : chr  "Microsoft Word" "Microsoft Word" "Microsoft Word"
"Microsoft Word" ...
```

- Now everything is a `character` instead of a `factor`

One-line Cleanup

- Let's clean up the `TVhours` column and cast it to numeric all in one command

```
survey <- transform(survey.messy, TVhours = as.numeric(gsub("[^0-9.]", "",
TVhours)))
str(survey)
```

```
'data.frame':   31 obs. of  6 variables:
 $ Program        : chr  "MISM" "Other" "MISM" "PPM" ...
 $ PriorExp       : chr  "Some experience" "Some experience" "Extensive
experience" "Never programmed before" ...
 $ Rexperience    : chr  "Never used" "Basic competence" "Basic competence"
"Installed on machine" ...
 $ OperatingSystem: chr  "Windows" "Windows" "Windows" "Windows" ...
 $ TVhours        : num  1 8 4 5 3 10 0 10 15 4 ...
 $ Editor         : chr  "Microsoft Word" "Microsoft Word" "Microsoft Word"
"Microsoft Word" ...
```

What about other character column?

```
table(survey[["Program"]])
```

```
 MISM Other    PPM
   10     6     15
```

```
table(as.factor(survey[["Program"]]))
```

```
 MISM Other    PPM
   10     6     15
```

- Having factors coded as characters may be OK for many parts of our analysis

Let’s fix it

```
# Figure out which columns are coded as characters
chr.indexes <- sapply(survey, FUN = is.character)
chr.indexes
```

```
        Program         PriorExp     Rexperience OperatingSystem
           TRUE             TRUE             TRUE            TRUE
       TVhours           Editor
          FALSE             TRUE
```

```
# Re-code all of the character columns to factors
survey[chr.indexes] <- lapply(survey[chr.indexes], FUN = as.factor)
```

Here is the outcome

```
str(survey)
```

```
'data.frame':   31 obs. of  6 variables:
 $ Program         : Factor w/ 3 levels "MISM","Other",..: 1 2 1 3 2 3 2 1 3 3 ...
 $ PriorExp        : Factor w/ 3 levels "Extensive experience",..: 3 3 1 2 2 3 3 3
2 1 ...
 $ Rexperience     : Factor w/ 3 levels "Basic competence",..: 3 1 1 2 3 3 3 3 3 3
...
 $ OperatingSystem: Factor w/ 2 levels "Mac OS X","Windows": 2 2 2 2 2 1 1 2 1 2
...
 $ TVhours         : num  1 8 4 5 3 10 0 10 15 4 ...
 $ Editor          : Factor w/ 1 level "Microsoft Word": 1 1 1 1 1 1 1 1 1 1 ...
```

- **Success!**

Loop

```
for(i in 1:4) {
  print(i)
}
```

```
[1] 1
[1] 2
[1] 3
[1] 4
```

```
phrase <- "Good Night, "
for(word in c("and", "Good", "Luck")) {
  phrase <- paste(phrase, word)
  print(phrase)
}
```

```
[1] "Good Night,  and"
[1] "Good Night,  and Good"
[1] "Good Night,  and Good Luck"
```

Loop Syntax

A **for loop** executes a chunk of code for every value of an **index variable** in an **index set**

- The basic syntax takes the form

```
for(index.variable in index.set) {
  code to be repeated at every value of index.variable
}
```

- The index set is often a vector of integers, but can be more general

Example

```
index.set <- rnorm(4)  # 4 random standard normal variables
index.set
```

```
[1] -1.9912818  0.7119512  0.3999575  0.5872188
```

```
for(i in index.set) {
  print(abs(i))
}
```

```
[1] 1.991282
[1] 0.7119512
[1] 0.3999575
[1] 0.5872188
```

Example

```
index.set <- list(name="Michael", weight=185, is.male=TRUE) # a list
for(i in index.set) {
  print(c(i, typeof(i)))
}
```

```
[1] "Michael"   "character"
[1] "185"    "double"
[1] "TRUE"    "logical"
```

Example

```
fake.data <- matrix(rnorm(500), ncol=5) # create fake 100 x 5 data set
head(fake.data,2) # print first two rows
```

```
          [,1]       [,2]       [,3]      [,4]      [,5]
[1,] 0.9551196 -0.3985364 -0.2338078 -1.018674 -0.684313
[2,] 0.6483192 -0.5487889  0.2091660  2.016935 -1.030668
```

```
col.sums <- numeric(ncol(fake.data)) # variable to store running column sums
for(i in 1:nrow(fake.data)) {
  col.sums <- col.sums + fake.data[i,] # add ith observation to the sum
}
print(col.sums)
```

```
[1] -12.808804 -11.552563   3.211727   4.049877  -9.901475
```

```
colSums(fake.data) # A better approach (see also colMeans())
```

```
[1] -12.808804 -11.552563   3.211727   4.049877  -9.901475
```

While loop

- **while loops** repeat a chunk of code while the specified condition remains true

```
day <- 1
num.days <- 365
while(day <= num.days) {
  day <- day + 1
}
```

- We won't really be using while loops in this class

Various apply() function

Command	Description
`apply(X, MARGIN, FUN)`	Obtain a vector/array/list by applying `FUN` along the specified `MARGIN` of an array or matrix `X`
`lapply(X, FUN)`	Obtain a list by applying `FUN` to the elements of a list `X`
`sapply(X, FUN)`	Simplified version of `lapply`. Returns a vector/array instead of list.
`tapply(X, INDEX, FUN)`	Obtain a table by applying `FUN` to each combination of the factors given in `INDEX`

- These functions are (good!) alternatives to loops
- They are typically *more efficient* than loops (often run considerably faster on large data sets)
- Take practice to get used to, but make analysis easier to debug and less prone to error when used effectively

Example: apply()

```
colMeans(fake.data)
```

```
[1] -0.12808804 -0.11552563  0.03211727  0.04049877 -0.09901475
```

```
apply(fake.data, MARGIN=2, FUN=mean) # MARGIN = 1 for rows, 2 for columns
```

```
[1] -0.12808804 -0.11552563  0.03211727  0.04049877 -0.09901475
```

```
# Function that calculates proportion of vector indexes that are > 0
propPositive <- function(x) mean(x > 0)
apply(fake.data, MARGIN=2, FUN=propPositive)
```

```
[1] 0.51 0.47 0.51 0.51 0.46
```

Example: lapply(), sapply()

```
lapply(survey, is.factor) # Returns a list
```

```
$Program
[1] TRUE

$PriorExp
[1] TRUE

$Rexperience
[1] TRUE

$OperatingSystem
[1] TRUE

$TVhours
[1] FALSE

$Editor
[1] TRUE
```

```
sapply(survey, FUN = is.factor) # Returns a vector with named elements
```

```
     Program      PriorExp  Rexperience OperatingSystem
        TRUE          TRUE          TRUE          TRUE
     TVhours        Editor
       FALSE          TRUE
```

Example: apply, lapply, sapply

```r
apply(cars, 2, FUN=mean) # Data frames are arrays
```

```
speed  dist
15.40 42.98
```

```r
lapply(cars, FUN=mean) # Data frames are also lists
```

```
$speed
[1] 15.4

$dist
[1] 42.98
```

```r
sapply(cars, FUN=mean) # sapply() is just simplified lapply()
```

```
speed  dist
15.40 42.98
```

tapply()

- Think of `tapply()` as a generalized form of the `table()` function

```
library(MASS)
# Get a count table, data broken down by Origin and DriveTrain
table(Cars93$Origin, Cars93$DriveTrain)
```

```
          4WD Front Rear
  USA       5    34    9
  non-USA   5    33    7
```

```
# Calculate average MPG.City, broken down by Origin and Drivetrain
tapply(Cars93$MPG.city, INDEX = Cars93[c("Origin", "DriveTrain")], FUN=mean)
```

```
          DriveTrain
Origin       4WD    Front     Rear
  USA      17.6 22.14706 18.33333
  non-USA 23.4 24.93939 19.14286
```

Example: tapply()

- Let's get the average horsepower by car `Origin` and `Type`

```
tapply(Cars93[["Horsepower"]], INDEX = Cars93[c("Origin", "Type")], FUN=mean)
```

```
          Type
Origin      Compact    Large  Midsize    Small   Sporty     Van
  USA      117.4286 179.4545 153.5000 89.42857 166.5000 158.40
  non-USA 141.5556       NA 189.4167 91.78571 151.6667 138.25
```

- What's that `NA` doing there?

```
any(Cars93$Origin == "non-USA" & Cars93$Type == "Large")
```

```
[1] FALSE
```

- None of the non-USA manufacturers produced Large cars!

with()

- Thus far we've repeatedly typed out the data frame name when referencing its columns
- This is because the data variables don't exist in our working environment
- Using **with**(`data, expr`) lets us specify that the code in `expr` should be evaluated in an environment that contains the elements of `data` as variables

```
with(Cars93, table(Origin, Type))
```

```
          Type
Origin     Compact Large Midsize Small Sporty Van
  USA            7    11      10     7      8   5
  non-USA        9     0      12    14      6   4
```

Example: with()

```
any(Cars93$Origin == "non-USA" & Cars93$Type == "Large")
```

```
[1] FALSE
```

```
with(Cars93, any(Origin == "non-USA" & Type == "Large")) # Same effect!
```

```
[1] FALSE
```

```
with(Cars93, tapply(Horsepower, INDEX = list(Origin, Type), FUN=mean))
```

```
         Compact    Large  Midsize    Small   Sporty    Van
USA     117.4286 179.4545 153.5000 89.42857 166.5000 158.40
non-USA 141.5556       NA 189.4167 91.78571 151.6667 138.25
```

- Using with() makes code simpler, easier to read, and easier to debug

Workshop 6: Loop

Loop practice

(a) Write a function called calculateRowMeans that uses a for loop to calculate the row means of a matrix x.

(b) Try out your function on the random matrix fake.data defined below.

(b) Use the apply() function to calculate the row means of the matrix fake.data

(c) Compare this to the output of the rowMeans() function to check that your calculation is correct.

Other Packages

To load data

RODBC, RMySQL, RPostgresSQL, RSQLite - If you'd like to read in data from a database, these packages are a good place to start. Choose the package that fits your type of database.

XLConnect, xlsx - These packages help you read and write Micorsoft Excel files from R. You can also just export your spreadsheets from Excel as .csv's.

foreign - Want to read a SAS data set into R? Or an SPSS data set? Foreign provides functions that help you load data files from other programs into R.

R can handle plain text files – no package required. Just use the functions read.csv, read.table, and read.fwf..

xlsx

```
df <- read.xlsx("<name and extension of your file>",
                sheetIndex = 1)
```

Note that it is necessary to add a sheet name or a sheet index to this function. In the example above, the first sheet of the Excel file was assigned. If you have a bigger data set, you might get better performance when using the `read.xlsx2()` function:

```
df <- read.xlsx2("<name and extension of your file>",
                 sheetIndex = 1,
                 startRow=2,
                 colIndex = 2)
```

```
write.xlsx(df,
           "df.xlsx",
           sheetName="Data Frame")
```

The function requires you first to specify what data frame you want to export. In the second argument, you specify the name of the file that you are outputting. If, however, you want to write the data frame to a file that already exists, you can execute the following command:

```
write.xlsx(df,
           "<name and extension of your existing file>",
           sheetName="Data Frame"
           append=TRUE)
```

To manipulate data

dplyr - Essential shortcuts for subsetting, summarizing, rearranging, and joining together data sets. dplyr is our go to package for fast data manipulation.

tidyr - Tools for changing the layout of your data sets. Use the gather and spread functions to convert your data into the tidy format, the layout R likes best.

stringr - Easy to learn tools for regular expressions and character strings.

lubridate - Tools that make working with dates and times easier.

dplyr

install.packages("dplyr")

library(dplyr)

dplyr verbs	Description
`select()`	select columns
`filter()`	filter rows
`arrange()`	re-order or arrange rows
`mutate()`	create new columns
`summarise()`	summarise values
`group_by()`	allows for group operations in the "split-apply-combine" concept

select

Select a set of columns: the name and the sleep_total columns.

```
sleepData <- select(msleep, name, sleep_total)
head(sleepData)
```

```
##                          name sleep_total
## 1                     Cheetah        12.1
## 2                  Owl monkey        17.0
## 3              Mountain beaver        14.4
## 4 Greater short-tailed shrew        14.9
## 5                         Cow         4.0
## 6             Three-toed sloth        14.4
```

Subtraction (-)

To select all the columns *except* a specific column, use the "-" (subtraction) operator (also known as negative indexing)

```
head(select(msleep, -name))
```

```
##        genus  vore        order conservation sleep_total sleep_rem
## 1   Acinonyx carni    Carnivora           lc        12.1        NA
## 2      Aotus  omni     Primates         <NA>        17.0       1.8
## 3 Aplodontia herbi     Rodentia           nt        14.4       2.4
## 4    Blarina  omni Soricomorpha           lc        14.9       2.3
## 5        Bos herbi Artiodactyla domesticated         4.0       0.7
## 6   Bradypus herbi       Pilosa         <NA>        14.4       2.2
##   sleep_cycle awake brainwt  bodywt
## 1          NA  11.9      NA  50.000
## 2          NA   7.0 0.01550   0.480
## 3          NA   9.6      NA   1.350
## 4   0.1333333   9.1 0.00029   0.019
## 5   0.6666667  20.0 0.42300 600.000
## 6   0.7666667   9.6      NA   3.850
```

colon (:)

To select a range of columns by name, use the ":" (colon) operator

```
head(select(msleep, name:order))
```

```
##                      name      genus  vore        order
## 1                 Cheetah   Acinonyx carni    Carnivora
## 2               Owl monkey      Aotus  omni     Primates
## 3         Mountain beaver Aplodontia herbi     Rodentia
## 4 Greater short-tailed shrew    Blarina  omni Soricomorpha
## 5                     Cow        Bos herbi Artiodactyla
## 6           Three-toed sloth   Bradypus herbi       Pilosa
```

To select all columns that start with the character string “sl”, use the function `starts_with()`

```
head(select(msleep, starts_with("sl")))
```

```
##   sleep_total sleep_rem sleep_cycle
## 1        12.1        NA          NA
## 2        17.0       1.8          NA
## 3        14.4       2.4          NA
## 4        14.9       2.3   0.1333333
## 5         4.0       0.7   0.6666667
## 6        14.4       2.2   0.7666667
```

Some additional options to select columns based on a specific criteria include

1. `ends_with()` = Select columns that end with a character string
2. `contains()` = Select columns that contain a character string
3. `matches()` = Select columns that match a regular expression
4. `one_of()` = Select columns names that are from a group of names

filter

Filter the rows for mammals that sleep a total of more than 16 hours.

```
filter(msleep, sleep_total >= 16)
```

```
##                    name        genus    vore           order conservation
## 1             Owl monkey        Aotus    omni         Primates         <NA>
## 2   Long-nosed armadillo      Dasypus   carni       Cingulata           lc
## 3 North American Opossum    Didelphis    omni Didelphimorphia           lc
## 4          Big brown bat    Eptesicus insecti      Chiroptera           lc
## 5   Thick-tailed opposum   Lutreolina   carni Didelphimorphia           lc
## 6       Little brown bat       Myotis insecti      Chiroptera         <NA>
## 7        Giant armadillo   Priodontes insecti       Cingulata           en
## 8 Arctic ground squirrel Spermophilus   herbi        Rodentia           lc
##   sleep_total sleep_rem sleep_cycle awake brainwt bodywt
## 1        17.0       1.8          NA   7.0 0.01550  0.480
## 2        17.4       3.1   0.3833333   6.6 0.01080  3.500
## 3        18.0       4.9   0.3333333   6.0 0.00630  1.700
## 4        19.7       3.9   0.1166667   4.3 0.00030  0.023
## 5        19.4       6.6          NA   4.6      NA  0.370
## 6        19.9       2.0   0.2000000   4.1 0.00025  0.010
## 7        18.1       6.1          NA   5.9 0.08100 60.000
## 8        16.6        NA          NA   7.4 0.00570  0.920
```

filter with and

Filter the rows for mammals that sleep a total of more than 16 hours *and* have a body weight of greater than 1 kilogram.

```
filter(msleep, sleep_total >= 16, bodywt >= 1)
```

```
##                    name      genus    vore           order conservation
## 1   Long-nosed armadillo    Dasypus   carni       Cingulata           lc
## 2 North American Opossum  Didelphis    omni Didelphimorphia           lc
## 3        Giant armadillo Priodontes insecti       Cingulata           en
##   sleep_total sleep_rem sleep_cycle awake brainwt bodywt
## 1        17.4       3.1   0.3833333   6.6  0.0108    3.5
## 2        18.0       4.9   0.3333333   6.0  0.0063    1.7
## 3        18.1       6.1          NA   5.9  0.0810   60.0
```

Pipe (%>%)

```
head(select(msleep, name, sleep_total))
```

```
##                          name sleep_total
## 1                     Cheetah        12.1
## 2                  Owl monkey        17.0
## 3              Mountain beaver        14.4
## 4 Greater short-tailed shrew        14.9
## 5                         Cow         4.0
## 6           Three-toed sloth        14.4
```

```
msleep %>%
    select(name, sleep_total) %>%
    head
```

```
##                          name sleep_total
## 1                     Cheetah        12.1
## 2                  Owl monkey        17.0
## 3              Mountain beaver        14.4
## 4 Greater short-tailed shrew        14.9
## 5                         Cow         4.0
## 6           Three-toed sloth        14.4
```

sqldf

library(sqldf)

```
setwd()
crashes <- read.csv("crashes.csv")
roads <- read.csv("roads.csv")
head(crashes)
```

```
##   Year           Road N_Crashes Volume
## 1 1991 Interstate 65         25  40000
## 2 1992 Interstate 65         37  41000
## 3 1993 Interstate 65         45  45000
## 4 1994 Interstate 65         46  45600
## 5 1995 Interstate 65         46  49000
## 6 1996 Interstate 65         59  51000
```

```
tail(crashes)
```

```
##     Year           Road N_Crashes Volume
## 105 2007 Interstate 275         32  21900
## 106 2008 Interstate 275         21  21850
## 107 2009 Interstate 275         25  22100
## 108 2010 Interstate 275         24  21500
## 109 2011 Interstate 275         23  20300
## 110 2012 Interstate 275         22  21200
```

```
print(roads)
```

```
##             Road       District Length
## 1 Interstate 65     Greenfield    262
## 2 Interstate 70      Vincennes    156
## 3         US-36 Crawfordsville    139
## 4         US-40     Greenfield    150
## 5         US-52 Crawfordsville    172
```

Outer join

```
join_string <- "select
                  crashes.*
                , roads.District
                , roads.Length
                from crashes
                  left join roads
                  on crashes.Road = roads.Road"
```

```
crashes_join_roads <- sqldf(join_string,stringsAsFactors = FALSE)
```

```
## Loading required package: tcltk
```

```
head(crashes_join_roads)
```

```
##   Year          Road N_Crashes Volume   District Length
## 1 1991 Interstate 65         25  40000 Greenfield    262
## 2 1992 Interstate 65         37  41000 Greenfield    262
## 3 1993 Interstate 65         45  45000 Greenfield    262
## 4 1994 Interstate 65         46  45600 Greenfield    262
## 5 1995 Interstate 65         46  49000 Greenfield    262
## 6 1996 Interstate 65         59  51000 Greenfield    262
```

Inner join

By using an inner join, only matching rows will be kept.

```
join_string2 <- "select
                   crashes.*
                 , roads.District
                 , roads.Length
                 from crashes
                   inner join roads
                   on crashes.Road = roads.Road"
```

```
crashes_join_roads2 <- sqldf(join_string2, stringsAsFactors = FALSE)
head(crashes_join_roads2)
```

```
##   Year           Road N_Crashes Volume   District Length
## 1 1991 Interstate 65         25  40000 Greenfield    262
## 2 1992 Interstate 65         37  41000 Greenfield    262
## 3 1993 Interstate 65         45  45000 Greenfield    262
## 4 1994 Interstate 65         46  45600 Greenfield    262
## 5 1995 Interstate 65         46  49000 Greenfield    262
## 6 1996 Interstate 65         59  51000 Greenfield    262
```

```
tail(crashes_join_roads2)
```

```
##     Year  Road N_Crashes Volume       District Length
## 83 2007 US-36        49  24000 Crawfordsville    139
## 84 2008 US-36        52  24500 Crawfordsville    139
## 85 2009 US-36        55  24700 Crawfordsville    139
## 86 2010 US-36        35  23000 Crawfordsville    139
## 87 2011 US-36        33  21000 Crawfordsville    139
## 88 2012 US-36        31  20500 Crawfordsville    139
```

Where

```
join_string2 <- "select
                  crashes.*
                , roads.District
                , roads.Length
                  from crashes
                      inner join roads
                      on crashes.Road = roads.Road
                  where crashes.Road = 'US-40'"
crashes_join_roads4 <- sqldf(join_string2,stringsAsFactors = FALSE)
head(crashes_join_roads4)
```

```
##   Year  Road N_Crashes Volume   District Length
## 1 1991 US-40        46  21000 Greenfield    150
## 2 1992 US-40       101  21500 Greenfield    150
## 3 1993 US-40        76  23000 Greenfield    150
## 4 1994 US-40        72  21000 Greenfield    150
## 5 1995 US-40        75  24000 Greenfield    150
## 6 1996 US-40       136  23500 Greenfield    150
```

```
tail(crashes_join_roads4)
```

```
##    Year  Road N_Crashes Volume   District Length
## 17 2007 US-40        45  59500 Greenfield    150
## 18 2008 US-40        23  61000 Greenfield    150
## 19 2009 US-40        67  65000 Greenfield    150
## 20 2010 US-40       102  67000 Greenfield    150
## 21 2011 US-40        87  67500 Greenfield    150
## 22 2012 US-40        32  67500 Greenfield    150
```

To visualize data

ggplot2 - R's famous package for making beautiful graphics. ggplot2 lets you use the grammar of graphics to build layered, customizable plots.

ggvis - Interactive, web based graphics built with the grammar of graphics.

rgl - Interactive 3D visualizations with R

htmlwidgets - A fast way to build interactive (javascript based) visualizations with R. Packages that implement htmlwidgets include:

leaflet (maps)

dygraphs (time series)

DT (tables)

diagrammeR (diagrams)

network3D (network graphs)

threeJS (3D scatterplots and globes).

googleVis - Let's you use Google Chart tools to visualize data in R. Google Chart tools used to be called Gapminder, the graphing software Hans Rosling made famous in hie TED talk.

To model data

car - car's Anova function is popular for making type II and type III Anova tables.

mgcv - Generalized Additive Models

lme4/nlme - Linear and Non-linear mixed effects models

randomForest - Random forest methods from machine learning

multcomp - Tools for multiple comparison testing

vcd - Visualization tools and tests for categorical data

glmnet - Lasso and elastic-net regression methods with cross validation

survival - Tools for survival analysis

To report results

shiny - Easily make interactive, web apps with R. A perfect way to explore data and share findings with non-programmers.

R Markdown - The perfect workflow for reproducible reporting. Write R code in your markdown reports. When you run render, R Markdown will replace the code with its results and then export your report as an HTML, pdf, or MS Word document, or a HTML or pdf slideshow. The result? Automated reporting. R Markdown is integrated straight into RStudio.

xtable - The xtable function takes an R object (like a data frame) and

Build R Package

Why write an R package?

- To keep track of the miscellaneous R functions that you write and reuse.
- To distribute the data and software that accompany a paper.

Creating the package skeleton

1.New Project -> New Directory -> R Package

2.Pick the name for your package (only letters, numbers, and .), and where it'll live locally.

3.Check the "Create git repository" because why not.

4.Click on the DESCRIPTION file, and edit it.

5.Put your code files in the R/ subdirectory.

6.Load the package ("Build" tab, More -> Load all; or ⌘-⇧-L)

DESCRIPTION file

Package: veepack

Version: 0.1

Date: 2016-11-27

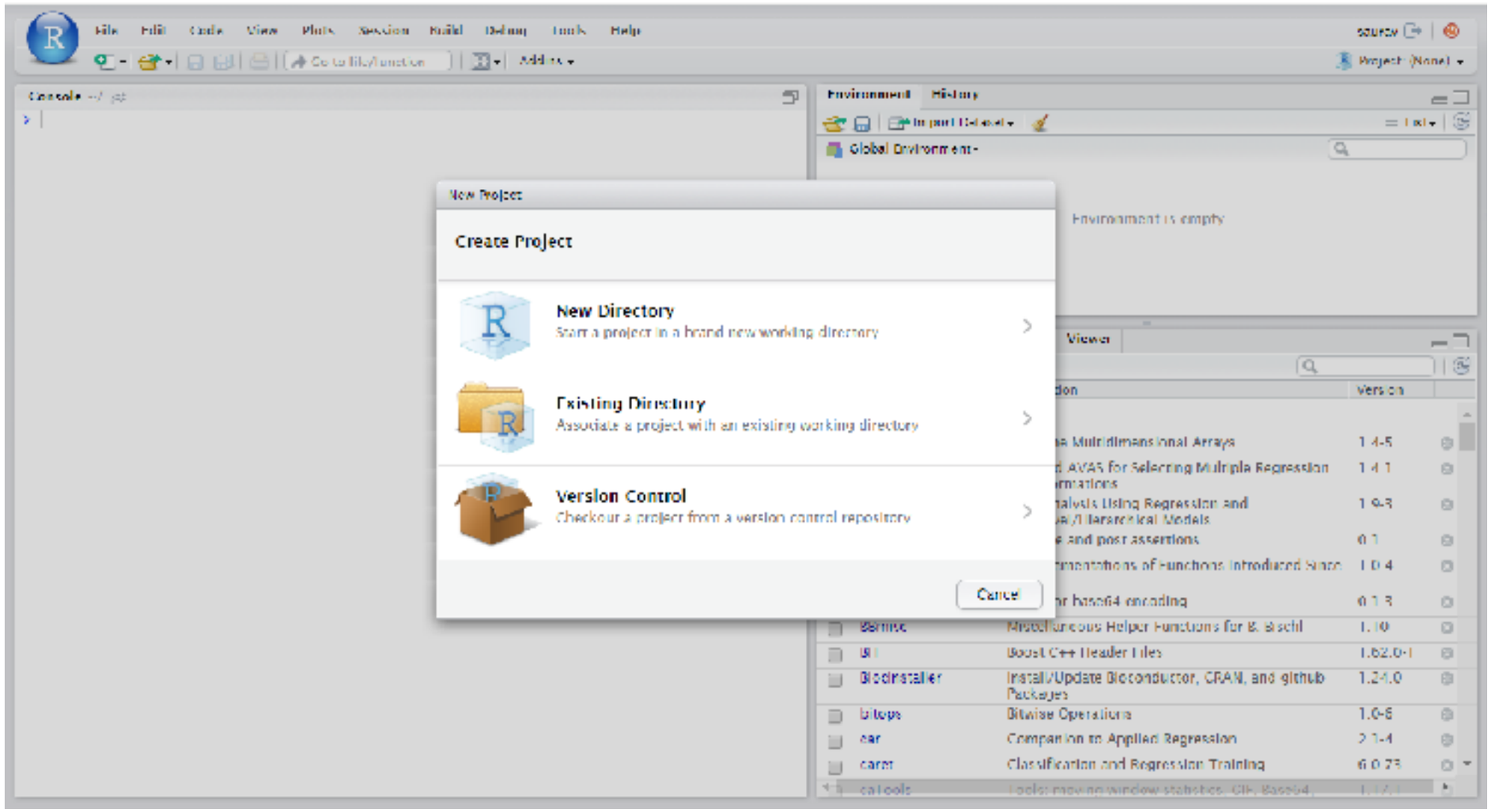
Title: Some functions

Description: Some description

Author: Veerasak Kritsanapraphan<veerasak.kritsanapraphan@gmail.com>

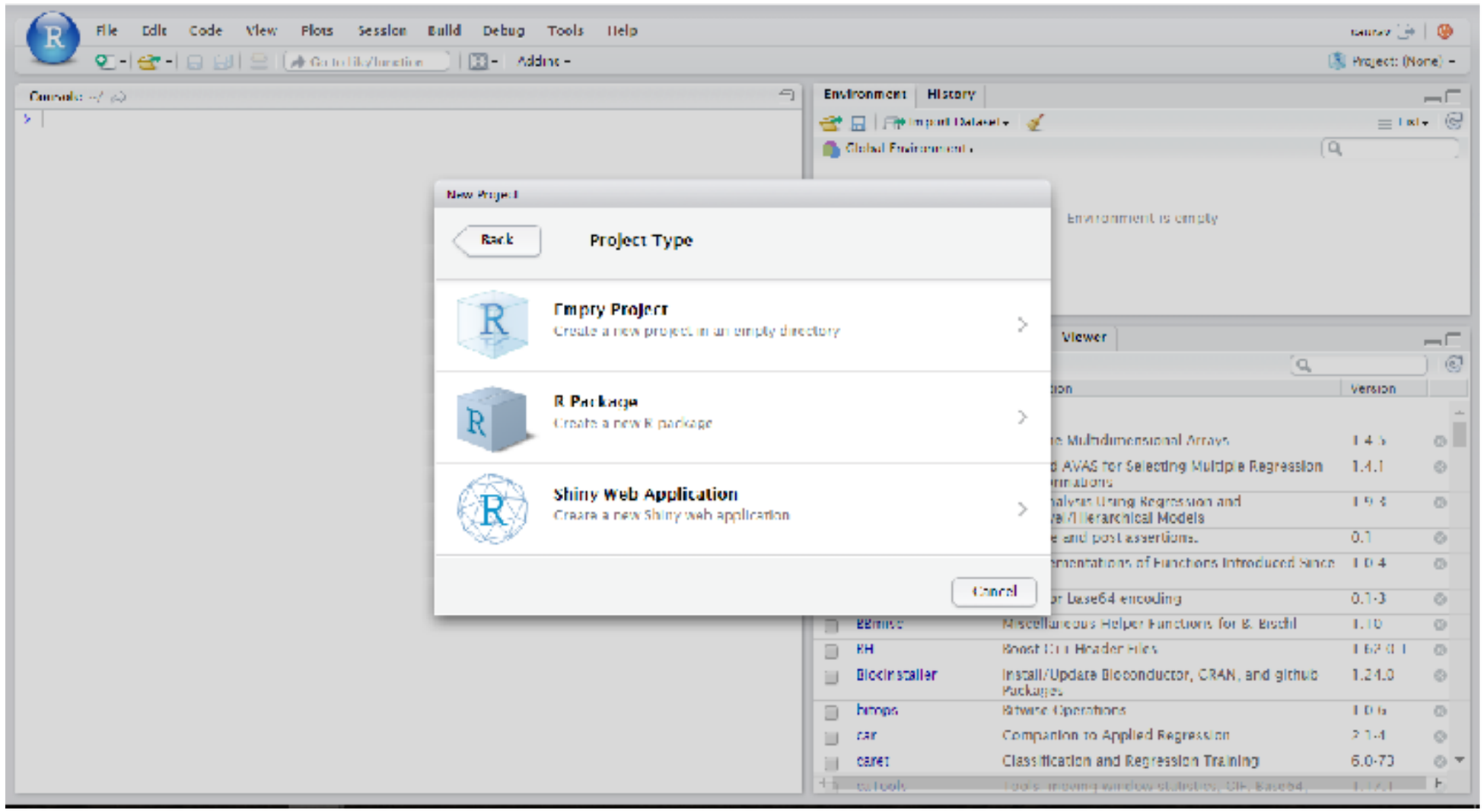
Maintainer: Veerasak Kritsanapraphan<veerasak.kritsanapraphan@gmail.com>

1.Create a new project by going to File > New Project.

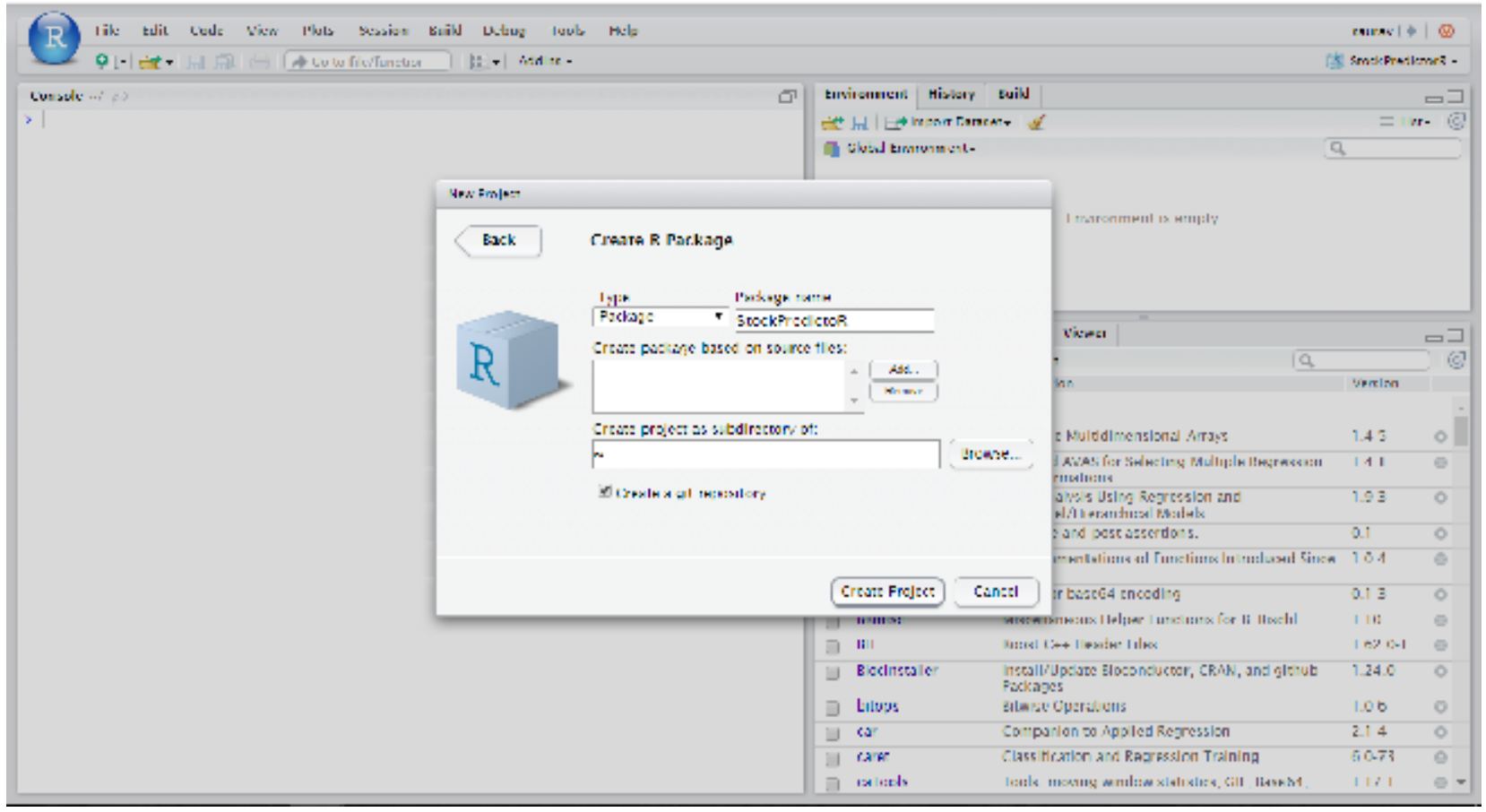


2. You can choose an existing directory or choose to create a new one.

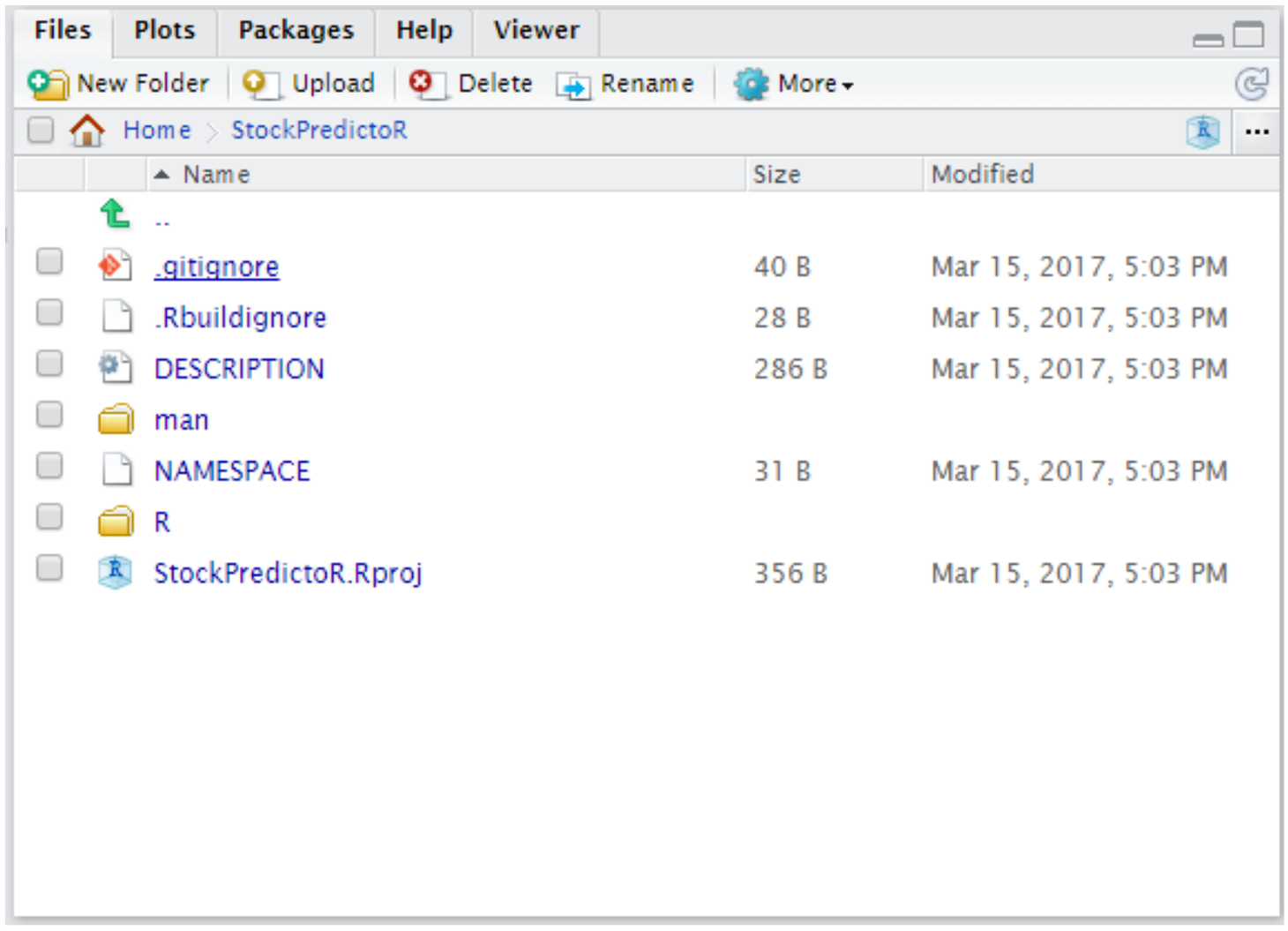
Then choose the project type as R package.



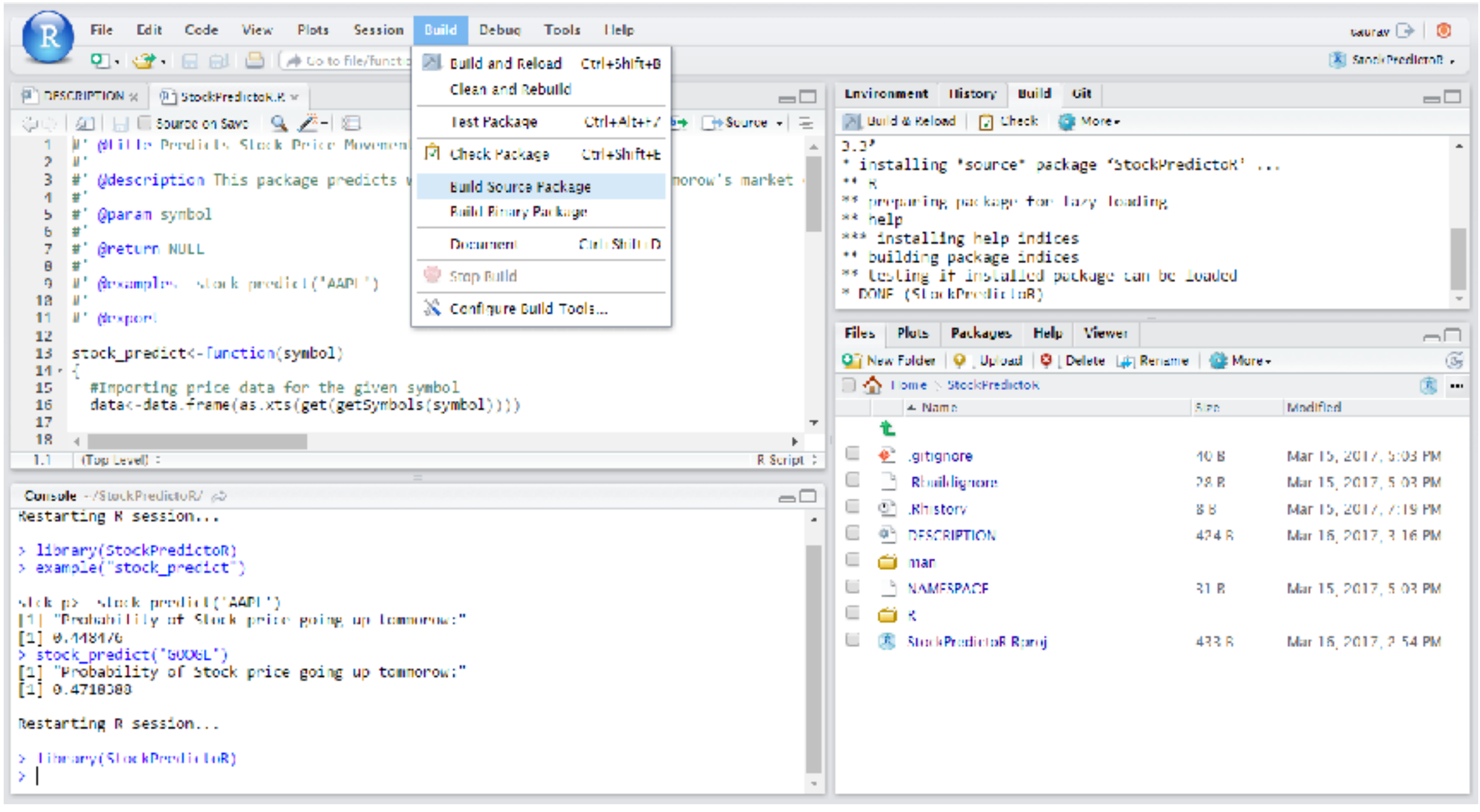
3. This is the time you choose an appropriate name for your package. I'm going to name it StockPredictoR. For naming your packages, you can use periods (like predictoR) or camel case as we are using here. I'll advice you to not use underscores while naming your packages. Also, choose the appropriate subdirectory that you want to store this project in.



4. This will create the following files in the directory. All the code will be stored in R folder while the manual and supporting documents will be stored in man folder.



5. Build package



Workshop 7: Building Packages

Try build your own package

A) Create your own Package name “addtwo”

B) Create function addTwo

C) Build and Install new package

End of Module 1