MONASH University

MONASH
INFORMATION
TECHNOLOGY

Introduction to Big Data

TP3 2018

GROUP OF EIGHT AUSTRALIA

Borhan Kazimipour (Brian Kazimi)

- **M.Sc. AI & ML**
  - Building predictive models on brain signals (EEG)
- **Ph.D. Optimization & ML**
  - Large-Scale Black-Box Optimization
- **GDDS Development**
  - FIT5148, FIT5196, FIT5197, FIT5201, FIT5202
- **GDDS Delivery (so far)**
  - FIT5148, FIT5196, FIT5201, FIT5202

- https://www.linkedin.com/in/borhan-kazimipour/
- https://scholar.google.com.au/citations?user=ZjCt2bkAAAAJ
- https://datascience.stackexchange.com/users/46505/borhan-kazimipour

Who? Me?

Shirin Ghaffarian Maghool

- **M.Sc. AI & ML**
  - Developing Movement Data Warehousing
- **GDDS Delivery (so far)**
  - FIT5147, FIT5148, FIT9133, FIT5202

...and also...

- **Happy**
  - to see familiar & new faces :)
- **Grateful**
  - because you are helping us improving the content
  - ... and because of your patience!
- **Excited**
  - as we will learn a lot from you as well

...and you!

- **Be active**
  - Ask your questions in the forum
  - Answer your friends questions
  - Share related resources on the forum
  - Participate in the discussion during meetups

- Do you know we have new venue?
  - Instead of email, please use Private Forum

Unit Intro

1. Introduction to Big Data
   a. intro to main concepts/definitions.
   b. the ecosystem of processing frameworks & data sources
   c. intro to popular prog. lang. Scala

2. Hadoop Ecosystem and Applications
   a. introduction to Apache Hadoop framework/architecture
   b. Hadoop distributed File System (HDFS)
   c. MapReduce prog. Model

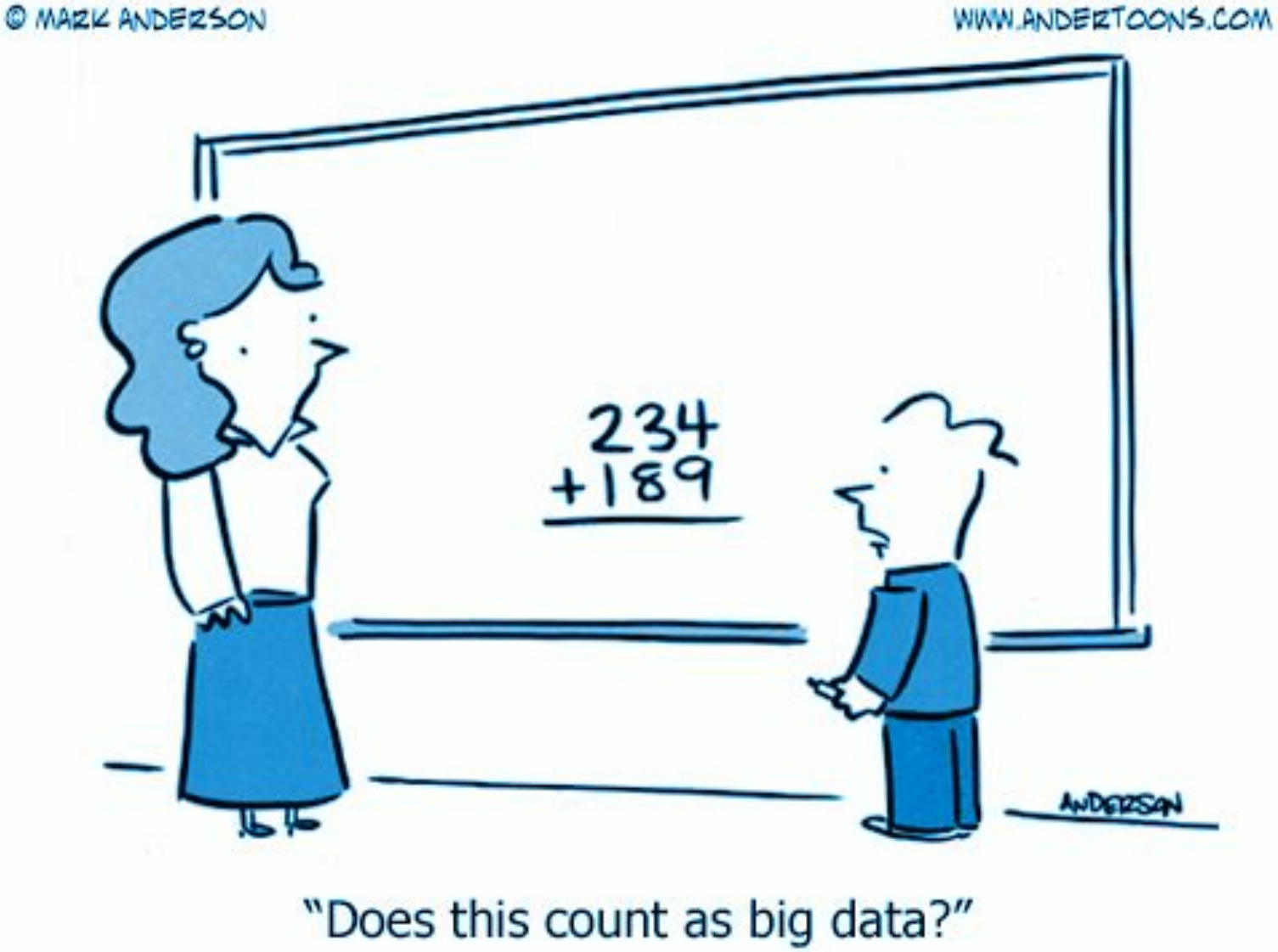
3. NoSQL 1: Big Data Processing Concepts and Technologies
   a. theory of distributed transactions & concurrency control.
   b. columnar type databases.
   c. Apache HBase architecture/operations

4. NoSQL 2: Spark & distributed graph processing
5. Data streams
6. Advanced Topics in Big Data

Unit Intro

1. Introduction to Big Data
2. Hadoop Ecosystem and Applications
3. NoSQL 1: Big Data Processing Concepts and Technologies
4. NoSQL 2: Spark & distributed graph processing
   a. intro to Apache Spark as a big data processing framework.
   b. Spark architecture/data abstractions/APIs/ETL
   c. intro to graph processing
   d. Spark GraphX
5. Data Streams
   a. intro to concepts in processing of continuously generated data.
   b. Time series, cash register, and turnstile data stream models.
   c. Sampling/sketching methods in Scala.
   d. Spark streaming module in Spark shell.
6. Advanced Topics in Big Data
   a. overview of main ML techniques for Big Dat.
   b. intro Spark MLLib
   c. intro data and compute clusters.

What is Big Data?



What is Big Data?

- Bernard Marr

  The **digital trace** that we are generating in this digital era.

- Lisa Arthur

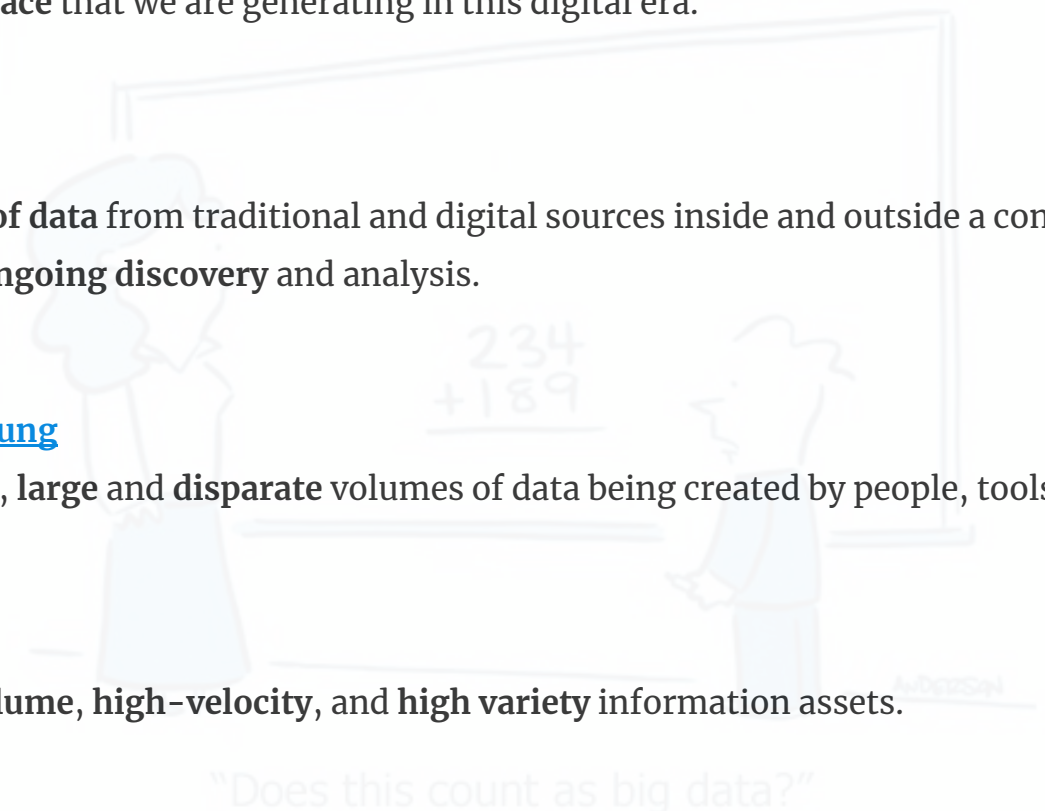
  A **collection of data** from traditional and digital sources inside and outside a company that represents a **source of ongoing discovery** and analysis.

- Ernst and Young

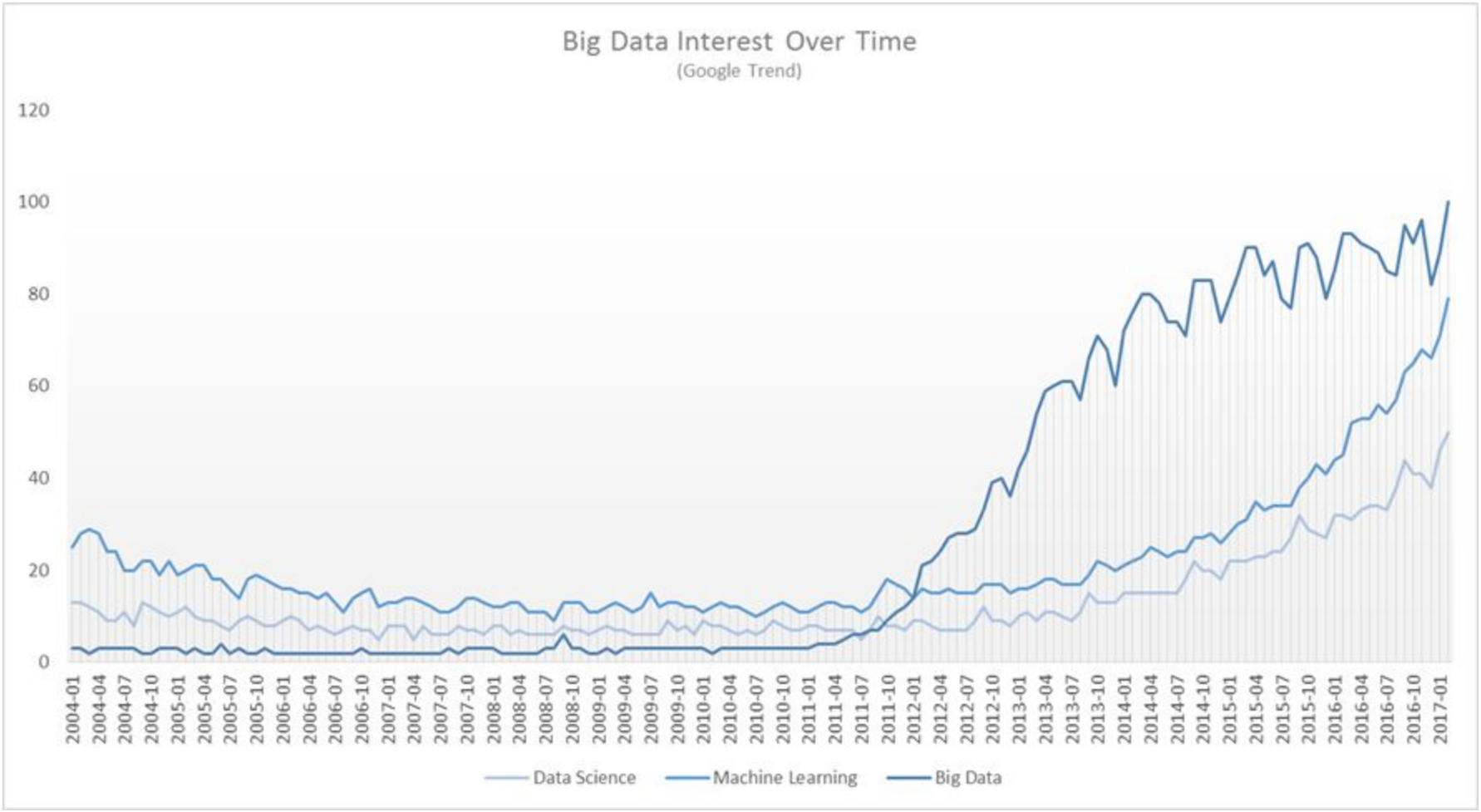
  The **dynamic**, **large** and **disparate** volumes of data being created by people, tools, and machines.

- Gartner

  The **high-volume**, **high-velocity**, and **high variety** information assets.

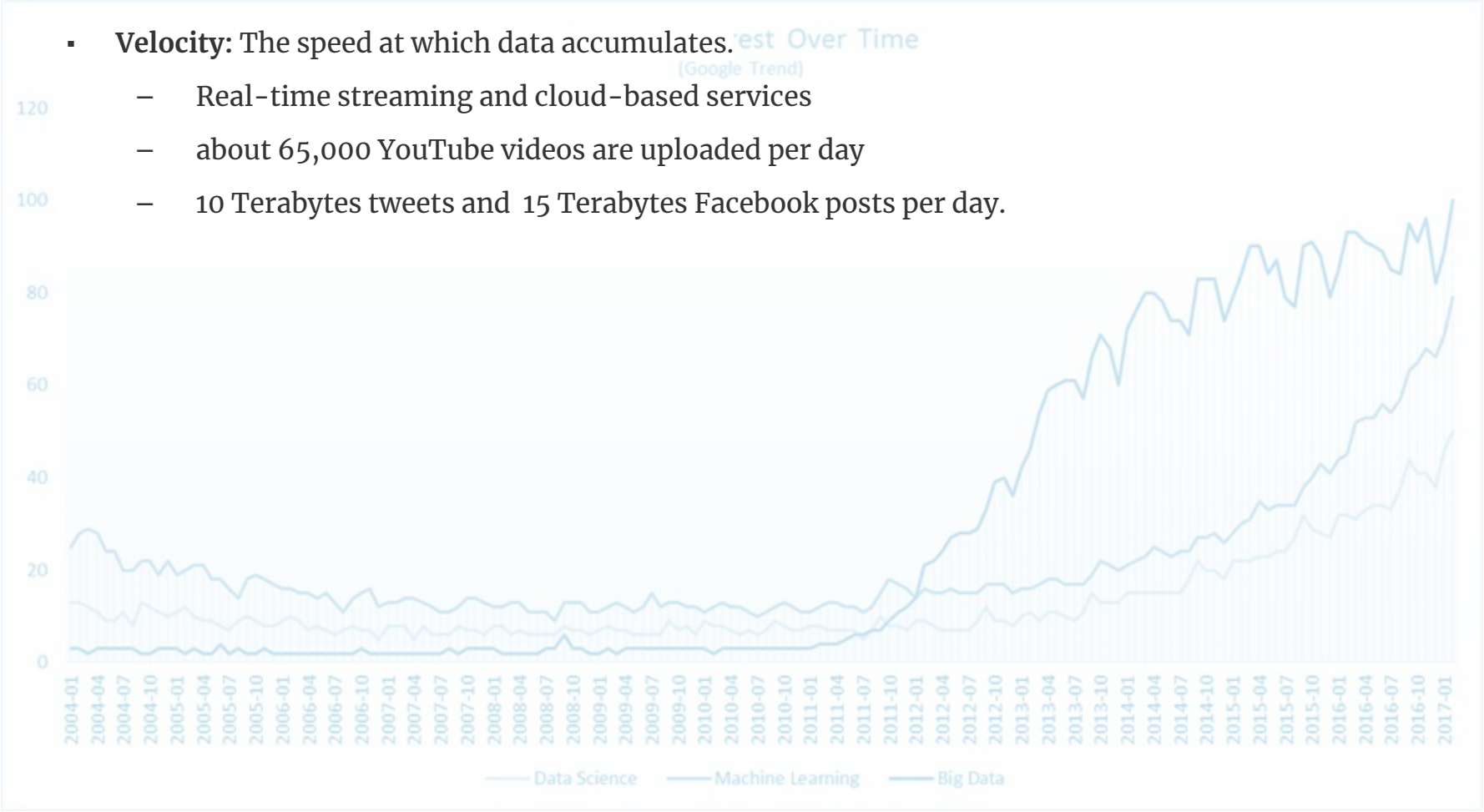


Big Data Characteristics



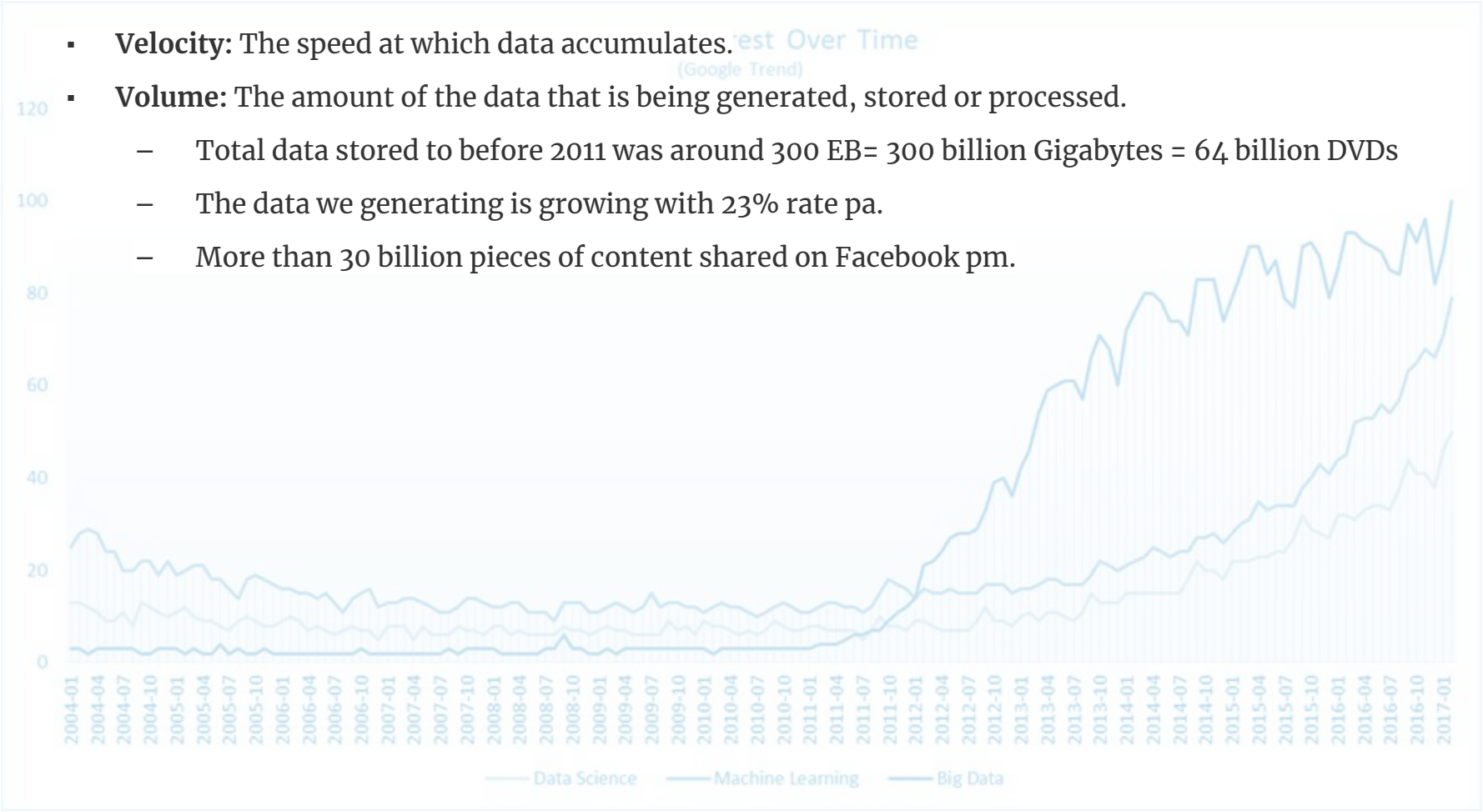
Big Data Characteristics

- **Velocity:** The speed at which data accumulates.
  - Real-time streaming and cloud-based services
  - about 65,000 YouTube videos are uploaded per day
  - 10 Terabytes tweets and 15 Terabytes Facebook posts per day.



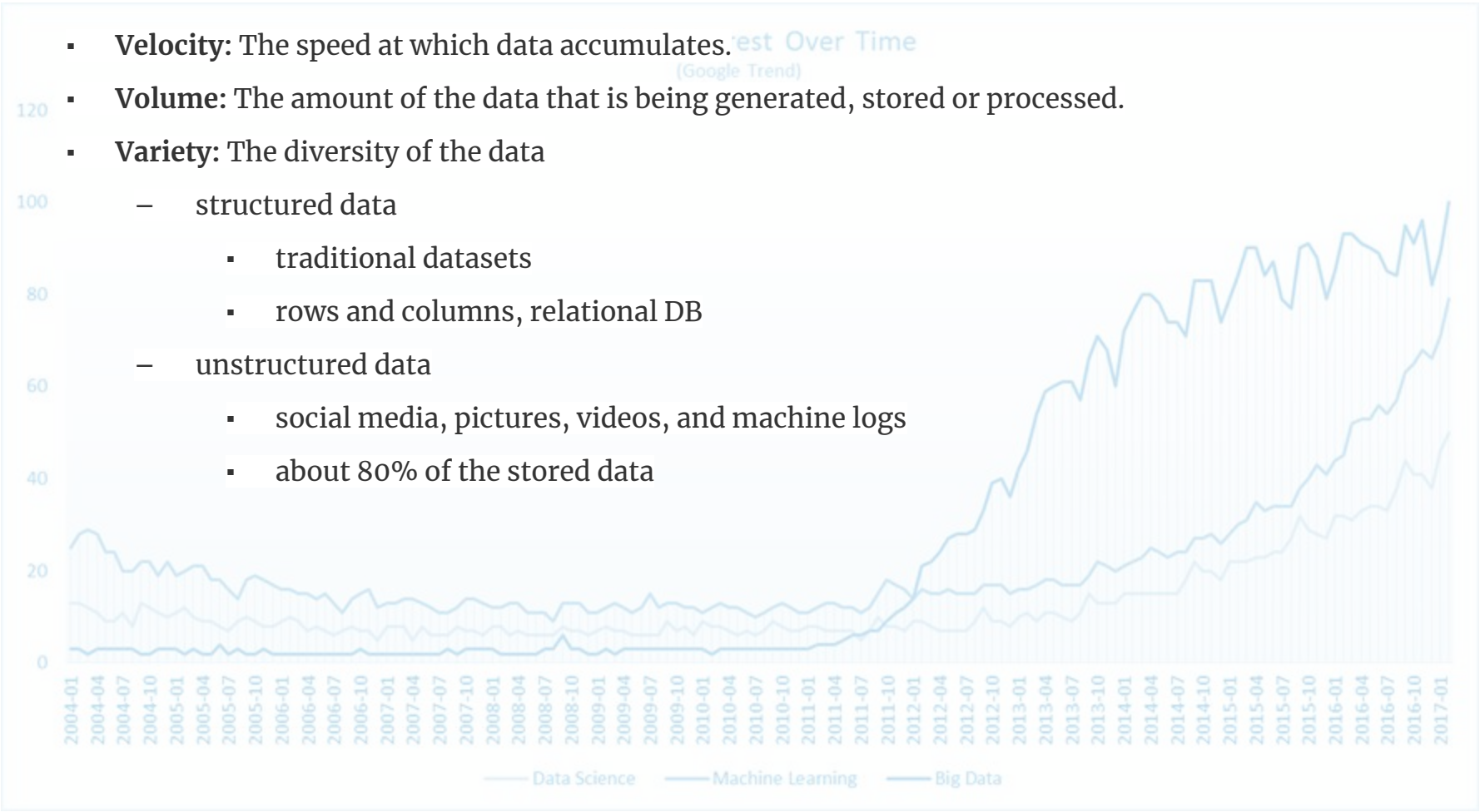
Big Data Characteristics

- **Velocity:** The speed at which data accumulates.
- **Volume:** The amount of the data that is being generated, stored or processed.
  - Total data stored to before 2011 was around 300 EB= 300 billion Gigabytes = 64 billion DVDs
  - The data we generating is growing with 23% rate pa.
  - More than 30 billion pieces of content shared on Facebook pm.



Big Data Characteristics

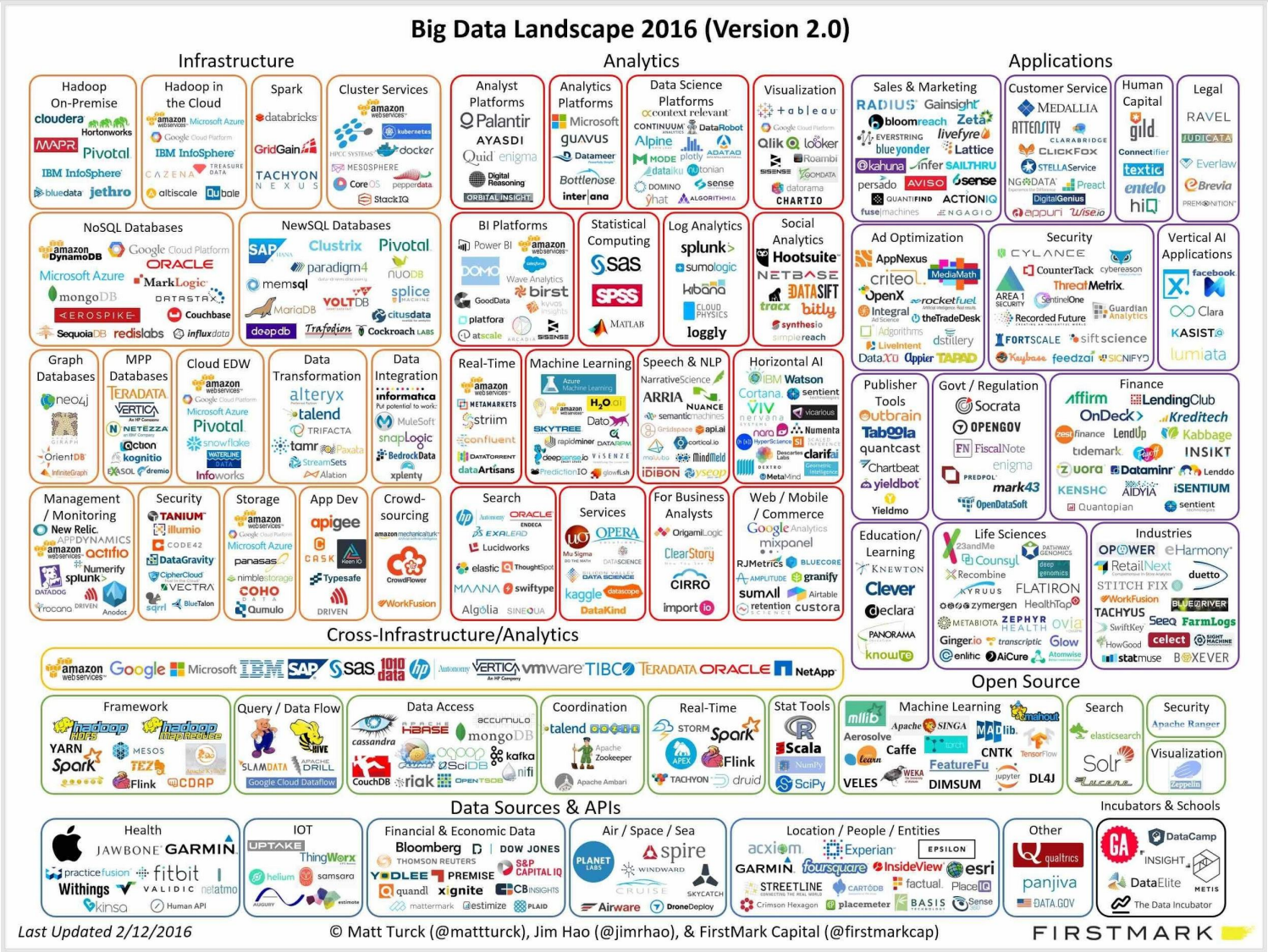
- **Velocity:** The speed at which data accumulates.
- **Volume:** The amount of the data that is being generated, stored or processed.
- **Variety:** The diversity of the data
  - structured data
    - traditional datasets
    - rows and columns, relational DB
  - unstructured data
    - social media, pictures, videos, and machine logs
    - about 80% of the stored data



Big Data Application

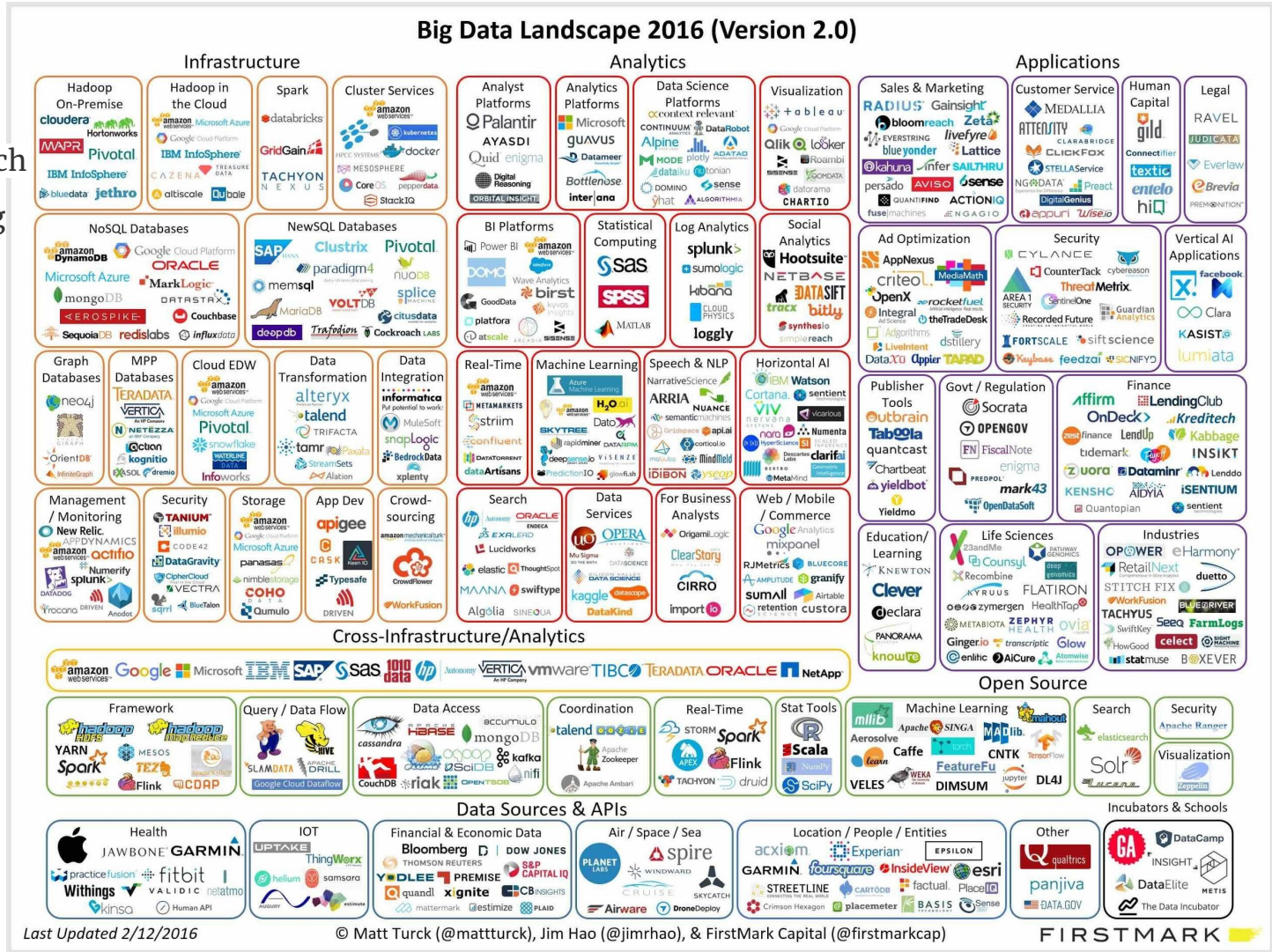
- **Some examples:**
  - Saving lives: Early detection of diseases.
  - Recommender systems/engines: Amazon, Google Adwords, and Spotify.
  - User behavior analysis: Predicting the successfulness of House of Cards series even before filming by Netflix
  - Question answering and summarization: Apple Siri and OK Google!
  - Scientific research: up to 2 billions human genomes could be sequenced by the year 2025!
  - Internet of Things (IoT): Reducing pollution and traffic congestion using traffic sensors, satellites, camera networks, smartphone GPS etc.
  - Saving wildlife

Ecosystem


Big Data Landscape 2016 (Version 2.0)
Infrastructure
Analytics
Applications
Hadoop On-Premise
Hadoop in the Cloud
Spark
Cluster Services
Analyst Platforms
Analytics Platforms
Data Science Platforms
Visualization
Sales & Marketing
Customer Service
Human Capital
Legal
NoSQL Databases
NewSQL Databases
BI Platforms
Statistical Computing
Log Analytics
Social Analytics
Ad Optimization
Security
Vertical AI Applications
Graph Databases
MPP Databases
Cloud EDW
Data Transformation
Data Integration
Real-Time
Machine Learning
Speech & NLP
Horizontal AI
Publisher Tools
Govt / Regulation
Finance
Management / Monitoring
Security
Storage
App Dev
Crowd-sourcing
Search
Data Services
For Business Analysts
Web / Mobile / Commerce
Education / Learning
Life Sciences
Industries
Cross-Infrastructure/Analytics
Open Source
Framework
Query / Data Flow
Data Access
Coordination
Real-Time
Stat Tools
Machine Learning
Search
Security
Visualization
Data Sources & APIs
Health
IOT
Financial & Economic Data
Air / Space / Sea
Location / People / Entities
Other
Incubators & Schools
Last Updated 2/12/2016
© Matt Turck (@mattturck), Jim Hao (@jimrhao), & FirstMark Capital (@firstmarkcap)
FIRSTMARK

Ecosystem

- **Data Sources**
  - Apple Watch
  - Bloomberg
  - Twitter
  - Vic Road
  - BOM
- **Infrastructures**
  - Hadoop
  - Spark
  - NoSQL
- **Analytics**
  - BigInsight
  - PowerBI
- **Visualization**
  - Tableau
  - Talend



Challenges

1. Understanding the data
2. Lack of experts
3. Right platform
4. Rapid changes
5. Data ownership
6. Security/privacy

Challenges

General Topics:

Your:

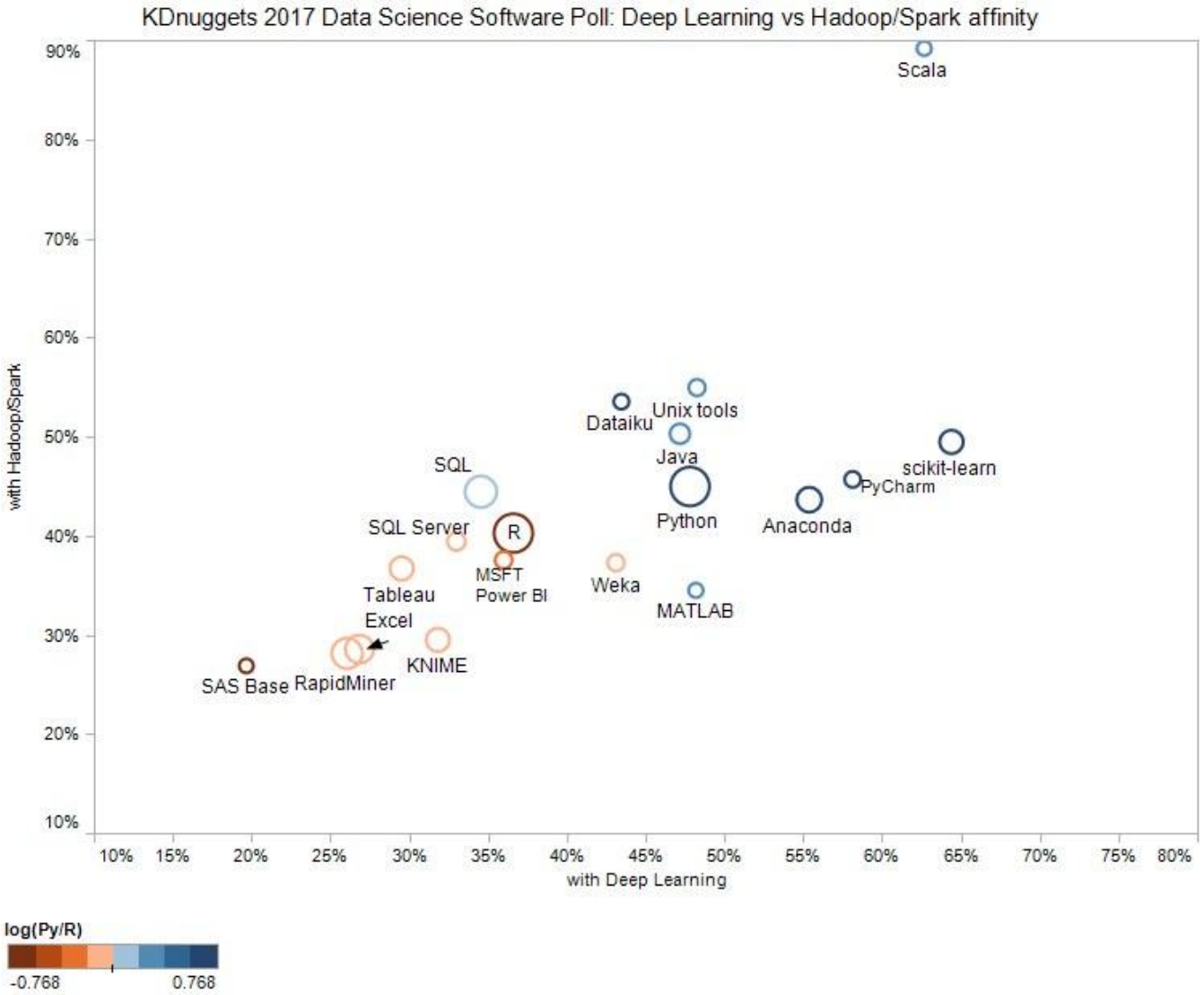
- name!
- background
- experience in Big Data
- favorite Big Data tool

Group Specific Topics:

Challenges in:

1. Understanding the data
2. Lack of experts
3. Choosing Right platform
4. Rapid changes
5. Data ownership
6. Security/privacy

Big Data Programming Languages



http://www.kdnuggets.com/aps/sw17-top11-dl-sh.anon.csv

Intro Scala

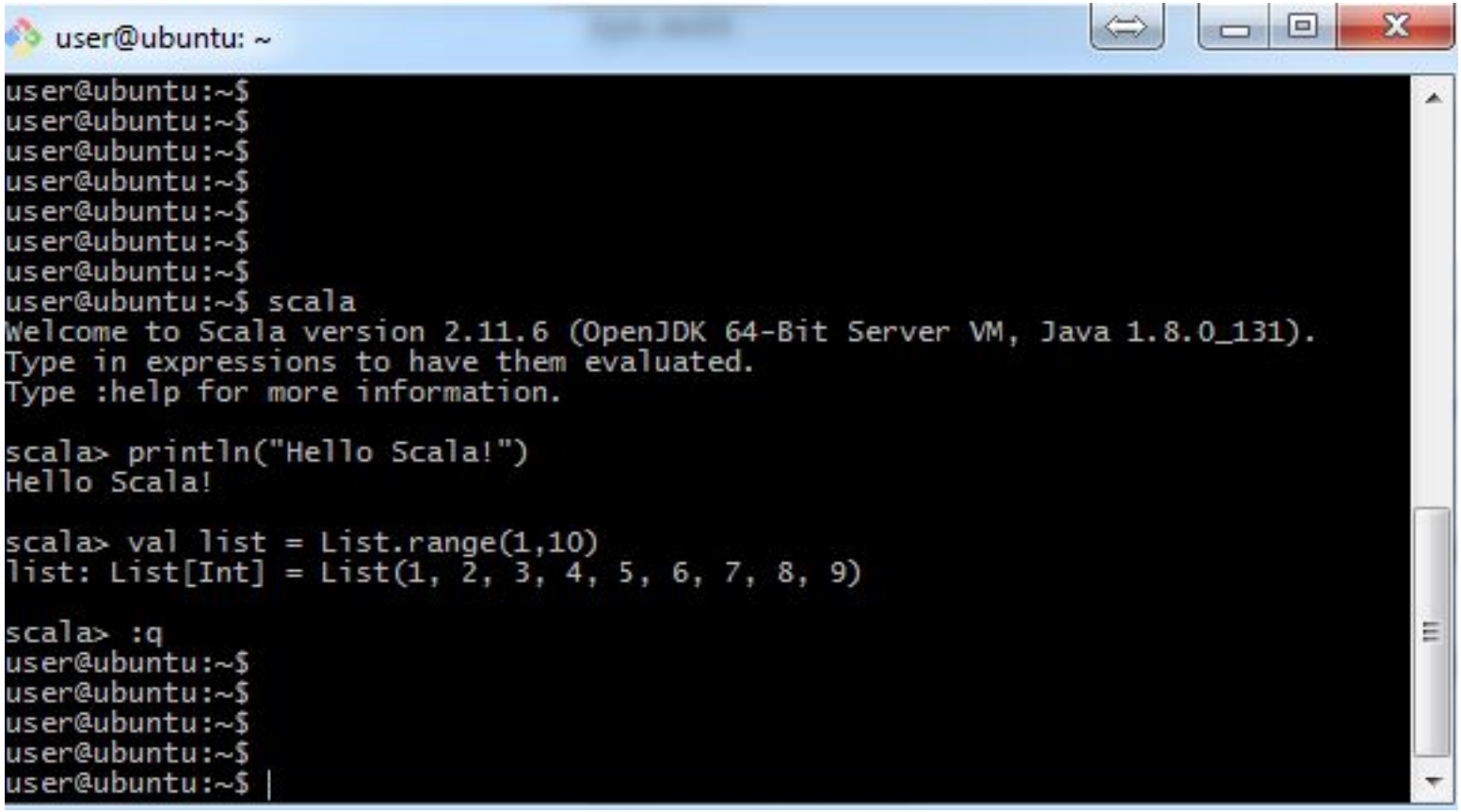
- **Why Scala**
  - Great integration with Hadoop & Spark
  - Scalability
  - Supportive community
  - Increasing popularity
- **Scala level**
  - You need to survive not master!
  - Knowing OOP (& Java) can be helpful
- **Environment**
  - Free to use any option for practice
  - Recommended: hosted Monash Big Data VM
- **IDE/GUI**
  - Free to pick & use any:
    - Jupyter
    - Zeppelin
    - scala REPL
    - scalac compiler (could be tricky)
    - Text editors: vi & nano (some learning curve)
    - Develop locally, upload to cloud, test, and redo!
    - Copy-paste!

Scala: where to start?

1. **Documentations** http://docs.scala-lang.org/
2. **Tutorials** http://people.cs.ksu.edu/~schmidt/705a/Scala/scala_tutorial.pdf
3. **Videos** https://www.youtube.com/watch?v=DzFt0YkZo8M
4. **Books**
   a. http://www.scala-lang.org/docu/files/ScalaByExample.pdf
   b. http://scalacookbook.com/
   c. http://alvinalexander.com/scala/scala-programming-cookbook-recipes-faqs
5. **Cheatsheets** http://alvinalexander.com/downloads/scala/Scala-Cheat-Sheet-devdaily.pdf
6. **Quick references** http://homepage.cs.uiowa.edu/~tinelli/classes/022/Fall13/Notes/scala-quick-reference.pdf
7. **Free online courses** https://www.coursera.org/courses?languages=en&query=scala

Hello Scala!

1. **Interactive Scala Programming**
   - scala
   - println("Hello, Scala!")



Hello Scala!

1. **Interactive Scala Programming**
   - scala
   - println("Hello, Scala!")

2. **Shell Script**
   - Create a .sh file
   - Use this preamble: #!/usr/bin/env scala
   - Write, save, change mode, and run.

3. **Compile and Run**
   - Create a .scala file
   - Compile using scalac compiler
   - Run using scala command

More: https://www.cis.upenn.edu/~matuszek/Concise%20Guides/Concise%20Scala.html

Scala: Data Types

- **Basics**
  - var or val?
    Quiz 1: What is the difference between val and var keywords?
  - Boolean, Int, Double, String, Unit, Null, Any,…
    Quiz 2: What is the difference between Unit and Null keywords?

    Quiz 3: Any? What is data type Any?
  - More: https://www.tutorialspoint.com/scala/scala_data_types.htm

Scala: Data Types

- **Basics**
  - var or val?
  - Boolean, Int, Double, String, Unit, Null, Any,...
  - More: https://www.tutorialspoint.com/scala/scala_data_types.htm

- **Extended**
  - Create new data using classes

```
1. class point(val x:Int, val y:Int)
2. {
3.   def move(dx: Int, dy: Int): point = {
4.     return new point(x + dx, y + dy)
5.   }
6.   override def toString(): String = "(" + x.toString() + "," + y.toString() +
")"
7. }
```

Scala: Data Types

```
1.  // define point class
2.  class point(val x:Int, val y:Int){
3.    def move(dx: Int, dy: Int) = new point(x+dx,y+dy)
4.    override def toString= "(" + x.toString + "," + y.toString + ")"
5.  }
6.  val pnt1 = new point(1,2) // create a point at x=1 and y=2
7.  println(pnt1) //print the point
8.  val pnt2 = pnt1.move(2,2) // move pnt1 2 units up and right
9.  println(pnt2) // print the second point
10. println(pnt1,pnt2) // print both points
```

Scala: Data Types

- **Basics**
  - var or val?
  - Boolean, Int, Double, String, Unit, Null, Any,...
  - More: https://www.tutorialspoint.com/scala/scala_data_types.htm

- **Extended**
  - Create new data using classes

```
1. class vector(end_point:point)
2. {
3.   override def toString: String = "< (0,0)" + "," + end_point.toString + ">"
4. }
```

Scala: Data Structure

- **Array**
  - concat(), empty(), fill(), range() and repeat().

```
1.  // Create and array of strings with length 10
2.  var anArray : Array[String] = new Array[String](10)
3.  // Does the same thing:
4.  var anotherArray  = new Array[String](10)
5.
6.  // Create an array of Int by initialization:
7.  var someNumbers = Array(1, 2, 3, 4, 5, 1, 2, 3, 4, 5)
8.
9.  // And now, a two dimensional array of size 10X5:
10. import Array._
11. val nRow = 10
12. val nCol = 5
13. var myMatrix = ofDim[Int](nRow,nCol)
```

Scala: Data Structure

- **Array**
  - concat(), empty(), fill(), range() and repeat().

```
1.  // Create and array of strings with length 10
2.  var anArray : Array[String] = new Array[String](10)
3.  // Does the same thing:
4.  var anotherArray  = new Array[String](10)
5.
6.  // Create an array of Int by initialization:
7.  var someNumbers = Array(1, 2, 3, 4, 5, 1, 2, 3, 4, 5)
8.
9.  // And now, a two dimensional array of size 10X5:
10. import Array._
11. val nRow = 10
12. val nCol = 5
13. var myMatrix = ofDim[Int](nRow,nCol)
```

Quiz: how to create an array of all odd positive numbers less than 1000*?*

Scala: Data Structure

- **List**
  - +, ::, :::, contains(), drop(), filter(), foreach(), last, min, max, and reverse.

```
1. // Create an empty String List
2. var aList : List[String] = List[String]()
3. // Does the same thing:
4. var anotherList  = List[String]()
5.
6. // Create an list of Int by initialization:
7. var someNumbers = List(1, 2, 3, 4, 5, 1, 2, 3, 4, 5)
```

```
1. // Recursively create a list of numbers:
2. var someNumbers = 1 :: (2 :: (3 :: (4 :: (5 :: (1 :: (2 :: (3 :: (4 :: (5 ::
   Nil)))))))))
```

Scala: Data Structure

- **List**
  - +, ::, :::, contains(), drop(), filter(), foreach(), last, min, max, and reverse.

```
1. // Create an empty String List
2. var aList : List[String] = List[String]()
3. // Does the same thing:
4. var anotherList  = List[String]()
5.
6. // Create an list of Int by initialization:
7. var someNumbers = List(1, 2, 3, 4, 5, 1, 2, 3, 4, 5)
```

```
1. // Recursively create a list of numbers:
2. var someNumbers = 1 :: (2 :: (3 :: (4 :: (5 :: (1 :: (2 :: (3 :: (4 :: (5 ::
   Nil)))))))))
```

Quiz: What ::: does?

Scala: Data Structure

- **Set**
  - +, -, ++, & , &~

```
1. val set1 = Set(5,10,15,20,25,30)
2. val set2 = Set(0,10,20,30,40)
3. val set3 = set1 ++ set2
```

Quiz 1: What is the difference between Set and Array?

Quiz 2: What is the difference between Set and Seq?

- **Tuple**
- **Map**

```
1. // Another way to create a color map
2. var colorMap:Map[String,String] = Map()
3.
4. colorMap += ("red" -> "FF0000")
5. colorMap += ("green" -> "00FF00")
6. colorMap += ("blue" -> "0000FF")
```

Scala: Functional Combinators

- **map**
  - def sqr(x: Int): Int = x * x
    val numList = List.range(1, 10)
    numList.map(sqr)
  - numList.map(_*2)
- **foreach**
  - numList.foreach(x => println(x*x))
  - numList.foreach(x => x*x)
- **filter**
  - numList.filter(_>5)
- **partition**
  - numList.partition(_>5)
- **zip**
  - numList.zip(numList.reverse)
- **find**
  - numList.find(_>5)
- **drop**
  - numList.drop(2)
- **dropWhile**
  - numList.dropWhile(_ % 2 != 0)

Scala: Functional Combinators

- **reduceLeft**
  - numList.reduceLeft(_+_)
- **reduceRight**
  - numList.reduceRight(_+_)

Any difference?

- **foldLeft and foldRight**
  - numList.foldLeft(100)(_+_)
- **flatten**
  - List(List(1, 2), List(3, 4)).flatten
- **flatMap**
  - Does map and then flatten!

Scala: Anything Else?

- **Anonymous function**
  - x => x+2
- **Element Operator**
  - numList.map(_+2)
  - numList.filter(_>0)
- **Type Alias**
  - type Set = Int => Boolean
- **Trait**
  - Traits are used to share interfaces and fields between different classes.
  - Think as superclasses that we can expand them to other classes (subtypes):

```
trait Pet { val name: String}

class Cat(val name: String) extends Pet

class Dog(val name: String) extends Pet

val dog = new Dog("Harry")

val cat = new Cat("Sally")
```

Sections 1.5.5-1.5-7 are optional!

Moodle Tour

MONASH
University

Any (simple) Question?


MONASH
University
GROUP
OF EIGHT
AUSTRALIA