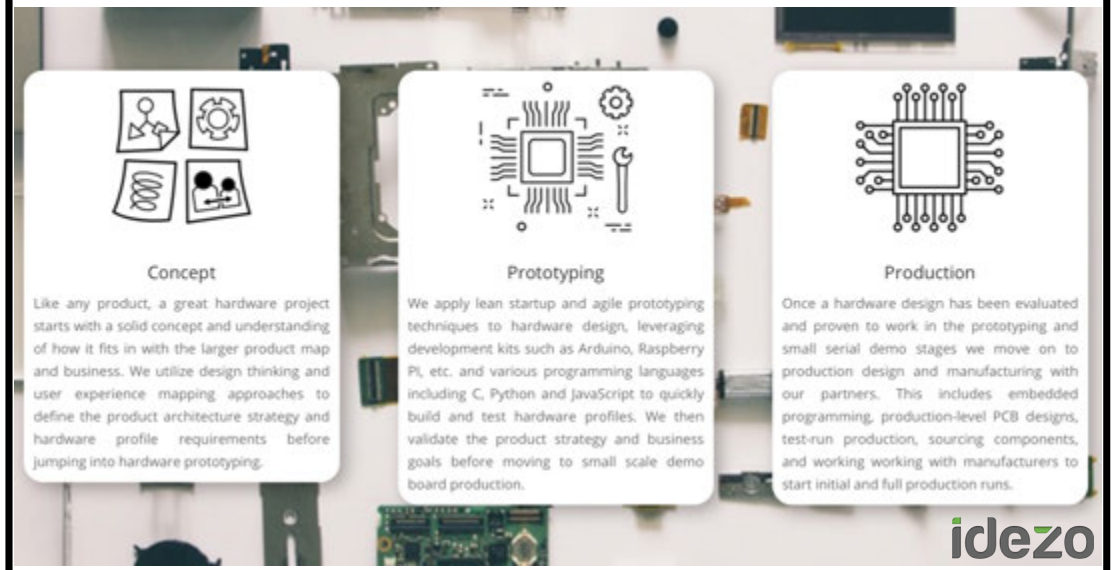
HANDS ON APPROACH TO

Data Science

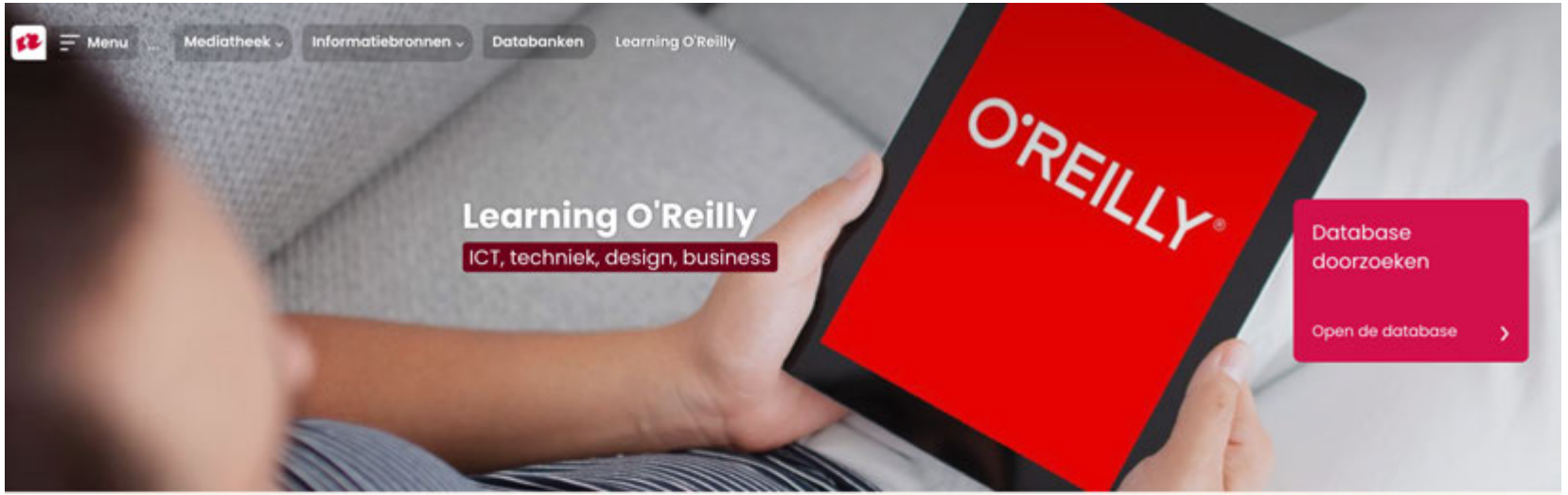
for (the)

IoT



Rob van der Willigen

https://github.com/robvdw/CMIDAT01K-DATA-SCIENCE-for-IOT



Learning O'Reilly is een Engelstalig online leerplatform gericht op ICT vaardigheden. Het biedt tienduizenden e-books en vele video's, case studies en "learning paths", waarbij je zelf je voortgang kunt monitoren.

Toegang

Via de website

De Learning O'Reilly website is gekoppeld aan de Hogeschool Rotterdam login. Kom je toch een O'Reilly inlogscherm tegen, vul dan alleen je @hr.nl e-mailadres in en klik op **Sign In with Single Sign On**. Krijg je een blanco pagina te zien? Schakel eventuele adblockers uit!

Via de O'Reilly app

Leer wat je wil, waar je wil. Met de O'Reilly app download je e-books en ga je verder met een playlist waar je op een ander apparaat gebleven was.

1 Google play App Store

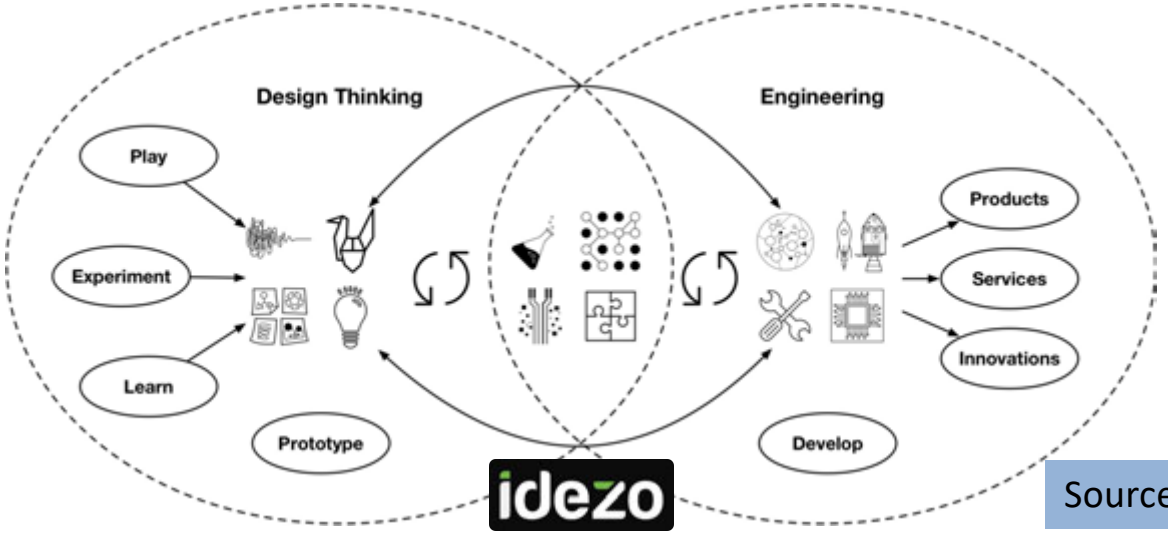
2 Gebruik je @hr.nl e-mailadres om in te loggen.

Inhoud

- Meer dan 8000 instructievideo's en 42.000 (hoofdzakelijk Engelstalige) e-books.

https://www.hogeschoolrotterdam.nl/voorlichting/voorzieningen/mediatheek/informatiebronnen/databases/mtdb/learning_oreilly/184698/

HANDS ON APPROACH TO DATA SCIENCE for (the) IoT



Source: https://idezo.ch

This Data Science Course was developed for Hogeschool Rotterdam **(Rotterdam University of Applied Sciences, RUAS) through the Program for AI & ethics (SPAiCE)**.
If you find errors or omissions, please contact the author, Rob van der Willigen, at r.f.van.der.willigen@hr.nl. Materials of this course and code examples used will become available at:
https://github.com/robvdw/CMIDAT01K-DATA-SCIENCE-for-IOT

lesson five

HOGESCHOOL ROTTERDAM

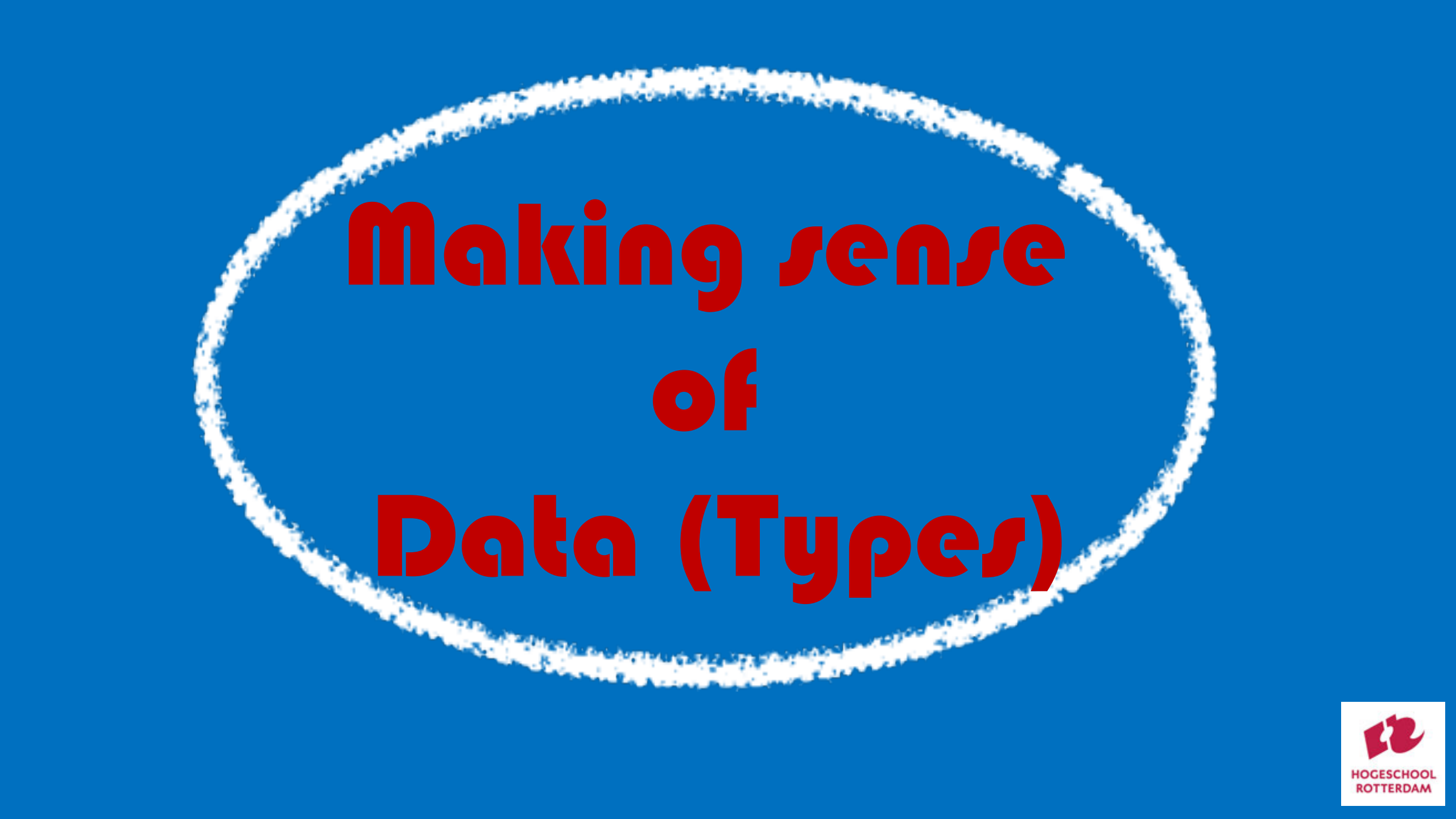
Course Setup

HOGESCHOOL ROTTERDAM

Lesson 01:	week 02	**Discovering the IoT Data Science Domain**
Lesson 02:	week 03	**Defining project requirements**
Lesson 03:	week 04	**Learn to write code**
Lesson 04:	week 05	**Data Science: How to start your own IoT Project**
Lesson 05:	week 07	**Data Types, IoT Platforms & MiddleWare**
Lesson 06:	week 08	**Core IoT Concepts + Code-testing via IoT** middelware
Lesson 07:	week 09	**Explaining Grading + Summary + Q & A**

Week 09:	**FEEDBACK**
Week 10:	**Submit your IoT-Project via LMS**

Making sense of Data (Types)



THE [Data Science] HYPE

There's huge and growing demand especially in business

Predicting the future take a lot of effort & expertise

Because of the hype, everyone wants to "own" data science!
- many of them are just selling their stuff with a new label

Here, I will ignore the hype and talk about Data
and how to use it for IoT-projects

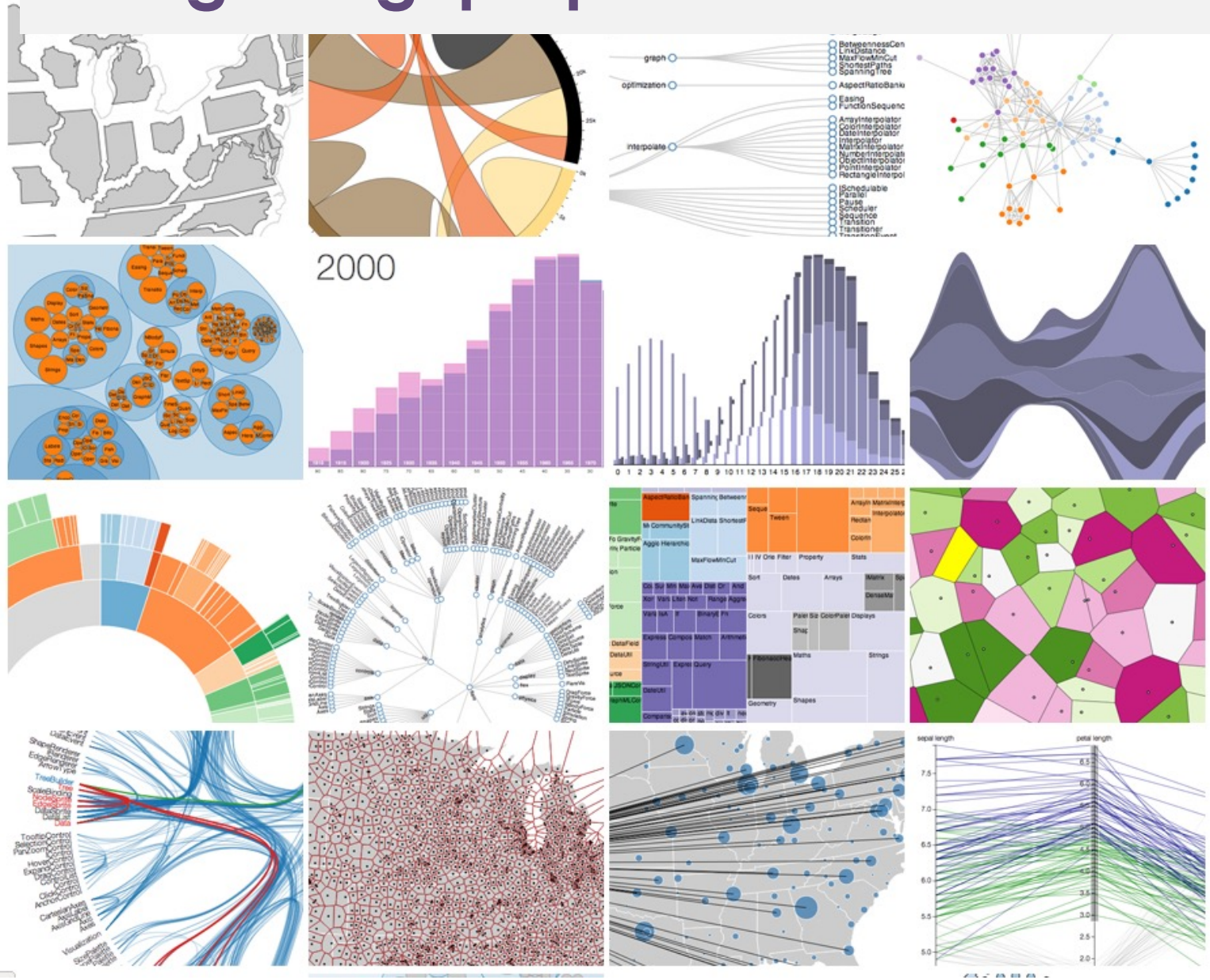
Data Science is about:

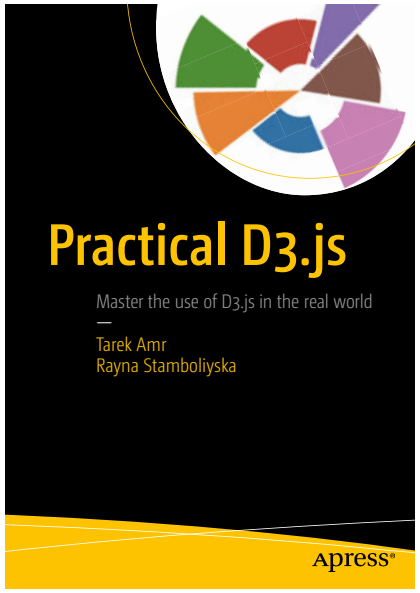
A Data Scientist can:

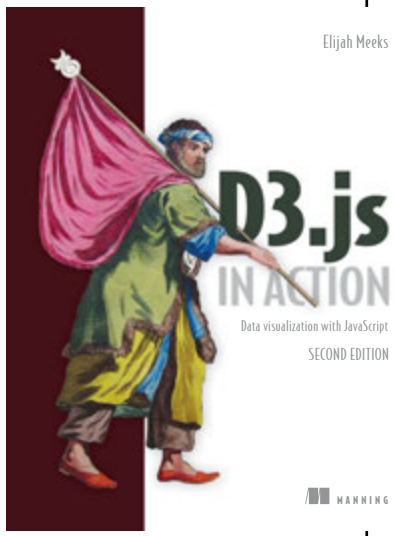
- *understand* the background domain
- *design* solutions that produce added value to the organization
- *implement* the solutions efficiently
- *communicate* the findings clearly (important!)

Data Scientist is a *practitioner* with sufficient expertise in software engineering, statistics/machine learning, **and** the application domain.

Wat geeft grip op DATA?



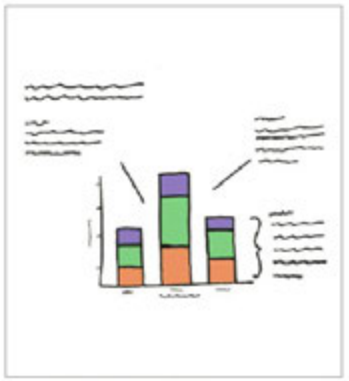




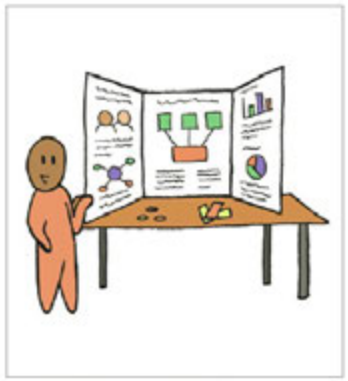
Data Types

Data VIZ ➔ Top-down **Cognition**
requires a Narrative (Story Telling)

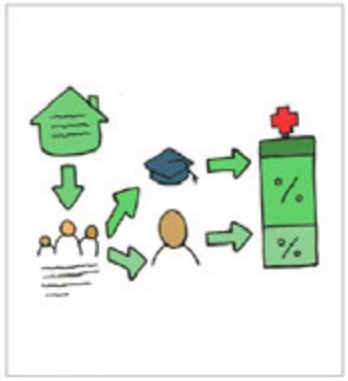
↳ Magazine style



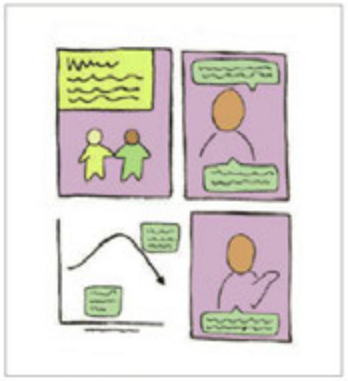
↳ Annotated chart



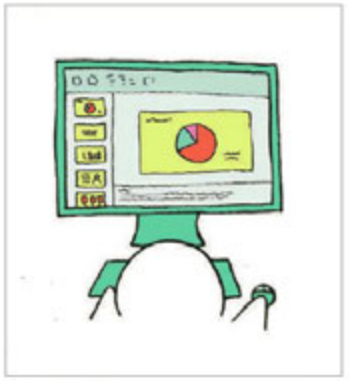
↳ Partitioned poster



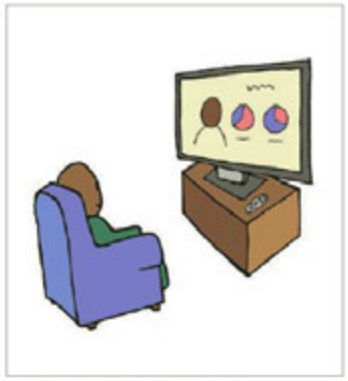
↳ Flow chart



↳ Comic strip



↳ Slide show



↳ Film/Video/Animation

DATA-DRIVEN IoT: WHAT IS DATA?

HOGESCHOOL ROTTERDAM

Data [gegevens]

Raw Facts
No Context
Numbers
Symbols

Data comes from the Latin word, "datum," meaning a "thing given."

Although the term "data" has been used since as early as the 1500s, modern usage started in the 1940s and 1950s as practical electronic computers began to input, process, and output data.

98734975471894614398734578
20875980542158009258202908
12349823094823048002343423

98734975471894614398734578
20875980542158009258202908
12349823094823048002343423

HOGESCHOOL ROTTERDAM

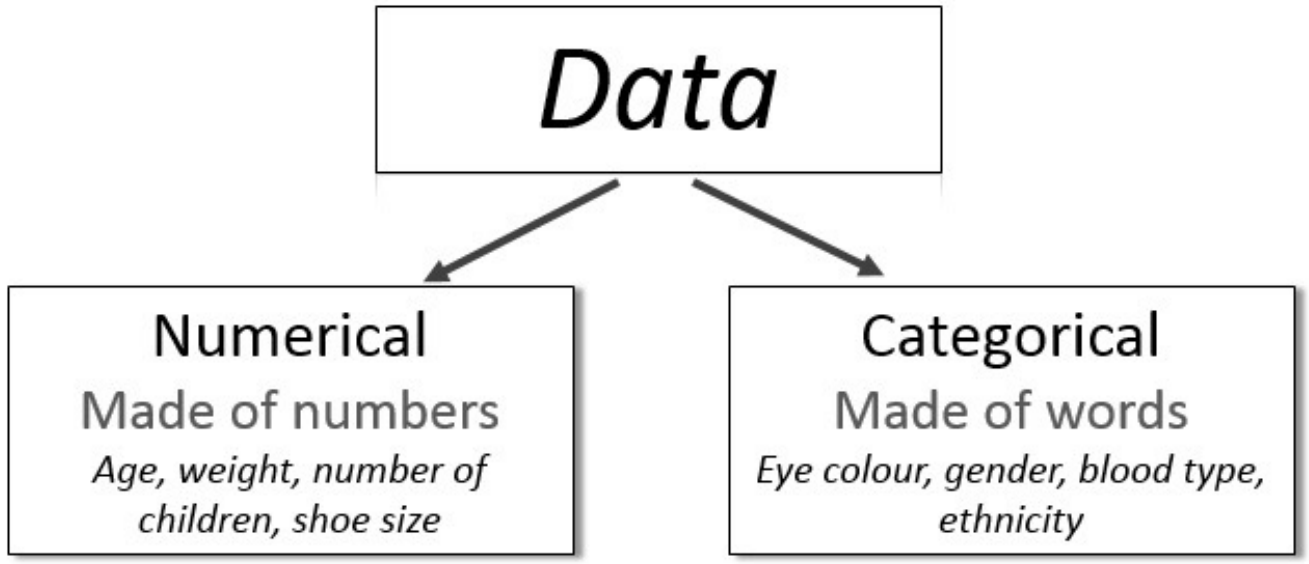
Can you find the the **mistake**?

1 2 3 4 5 6 7 8 9

TYPES OF DATA: Quantitative versus Qualitative [numerical vs categorical]

Data Quantification

Quantitative [Numerical] data: This data can be described using numbers, and basic mathematical procedures, including addition, are possible on the set. It can be discrete (countable numbers) or continuous (infinitely large or small)
Qualitative [Categorical] data: This data are categories. It cannot be described using numbers and basic mathematics. Is generally thought of as being described using"natural" categories and language.



- Quantitative values
  - **Measure** things
  - *Revenue, Units, Marketshare, Duration, Customer Satisfaction, Visits, Price, etc.*
- Categorical values
  - Subdivide things into **groups**
  - *Region, product, category, employee, etc.*

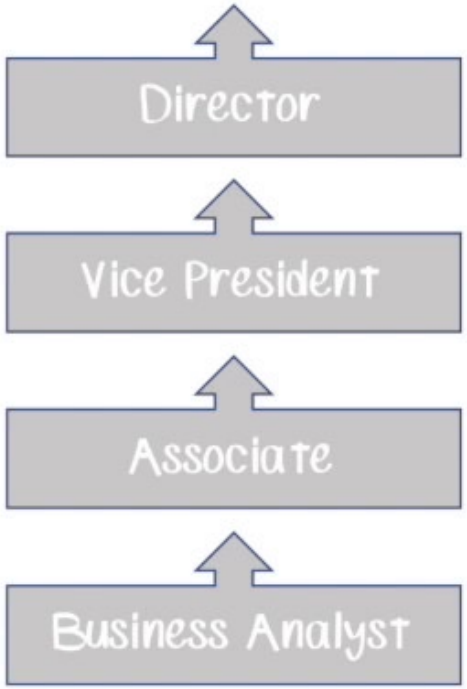

Data
Qualitative
Descriptive information
"I drink coffee every day"
Quantitative
Numerical information
Discrete
(Counted)
4
"I drink 4 coffees every day"
Continuous
(Measured)
80g
"I drink 80grs of coffee every day"

CATEGORICAL DATA

NOMINAL DATA

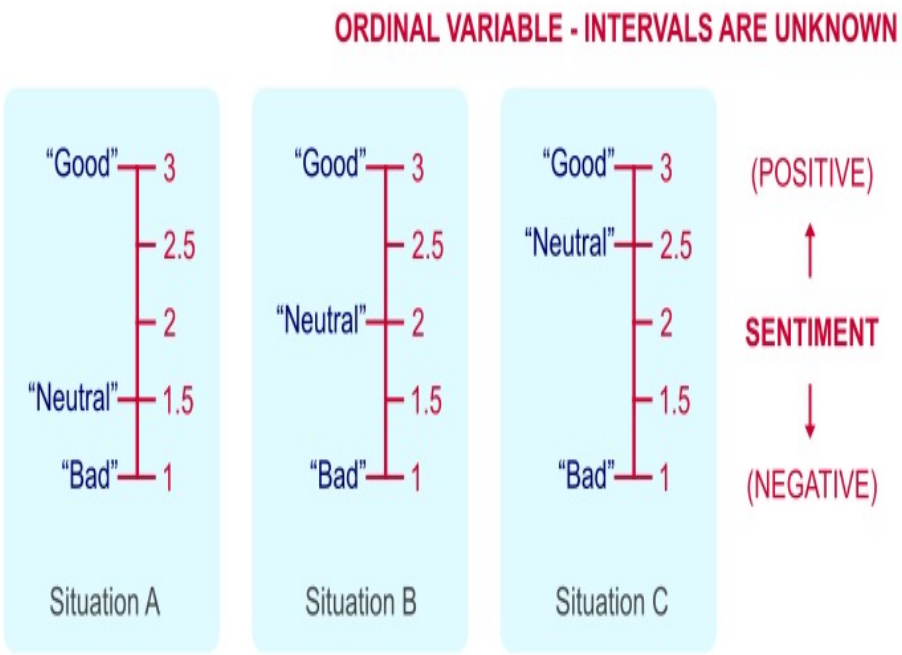
hair color

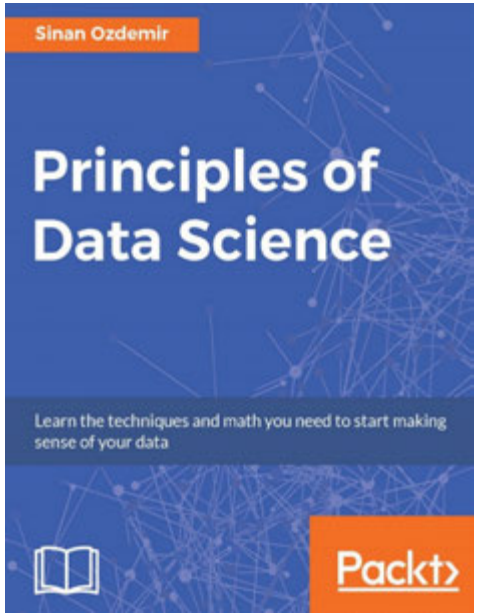
ORDINAL DATA



job hierarchy

Likert-scale data





Data

Numerical
Made of numbers

Age, weight, number of children, shoe size

Continuous
Infinite options

Age, weight, blood pressure

Discrete
Finite options

Shoe size, number of children

Categorical
Made of words

Eye colour, gender, blood type, ethnicity

Ordinal
Data has a hierarchy

Pain severity, satisfaction rating, mood

Nominal
Data has no hierarchy

Eye colour, dog breed, blood type

LEVELS OF DATA: LEVELS OF MEASUREMENTS/OBSERVATIONS

We onderscheiden 4 meetniveaus:

nominaal + ordinaal [discrete data]

interval + ratio [continue data]

Meetniveaus / Meetschalen:

Wanneer je onderzoek doet heb je vaak variabelen die je hierin moet verwerken.

Variabelen zijn elementen uit een onderzoek die verschillende waarden kunnen aannemen. Deze waarden kunnen worden gecategoriseerd in verschillende meetniveaus.

Meetniveaus kunnen iets vertellen over welke data-analyse geschikt is voor structurering.

LEVELS OF DATA: LEVELS OF MEASUREMENTS/OBSERVATIONS

Meetniveau	Wat je kunt berekenen met behulp van waarden op het meetniveau
Nominaal	Tellen, percentages berekenen
Ordinaal	Tellen, percentages berekenen en hoger/lager aangeven
Interval	Tellen, hoger/lager aangeven, verschillen in eenheden aangeven, gemiddelde, spreiding
Ratio	Tellen, hoger/lager aangeven, verschillen in eenheden aangeven, gemiddelde, spreiding en het berekenen van verhoudingen

LEVELS OF DATA: LEVELS OF MEASUREMENTS/OBSERVATIONS

Meetniveaus [level] /Meetschalen [scale]:

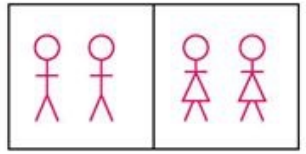
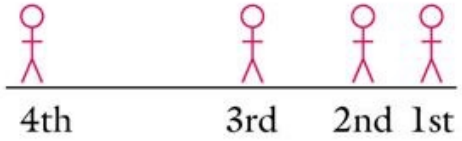
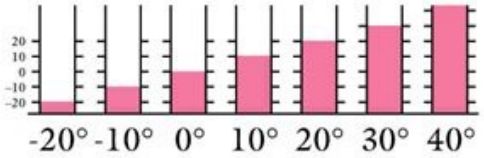
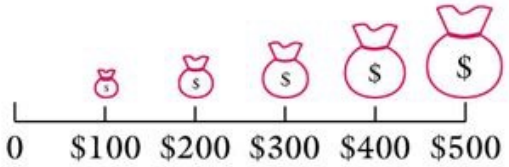
De hoogte van het meetniveau is bepalend voor:

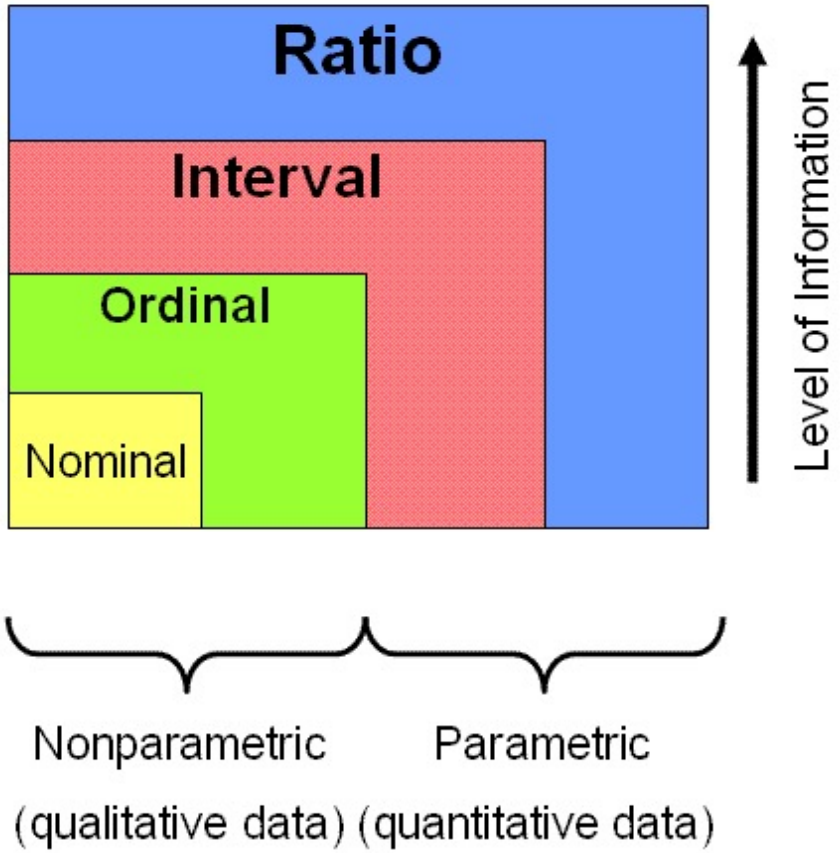
Statische-analyse + Grafische weergave

Meetniveaus & hun kenmerken

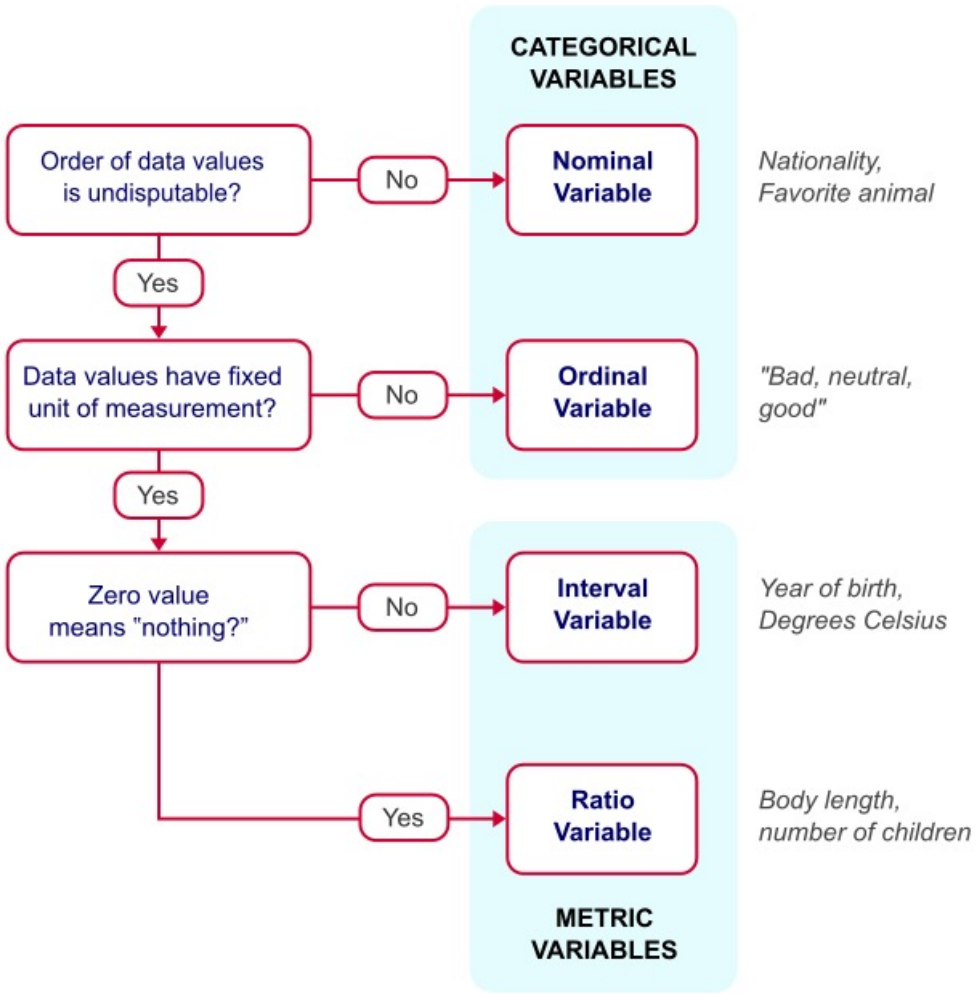
		Scale	
		Intervalniveau	**Rationiveau**
	Ordinaal niveau	Gelijke verschillen	Verhouding blijven gelijk
Nominale niveau	Ordening	Ordening	Gelijke verschillen
Onderscheid	Onderscheid	Onderscheid	Ordening
Geslacht	*Opleidingsniveau*	*Intelligentie*	Onderscheid
			Leeftijd

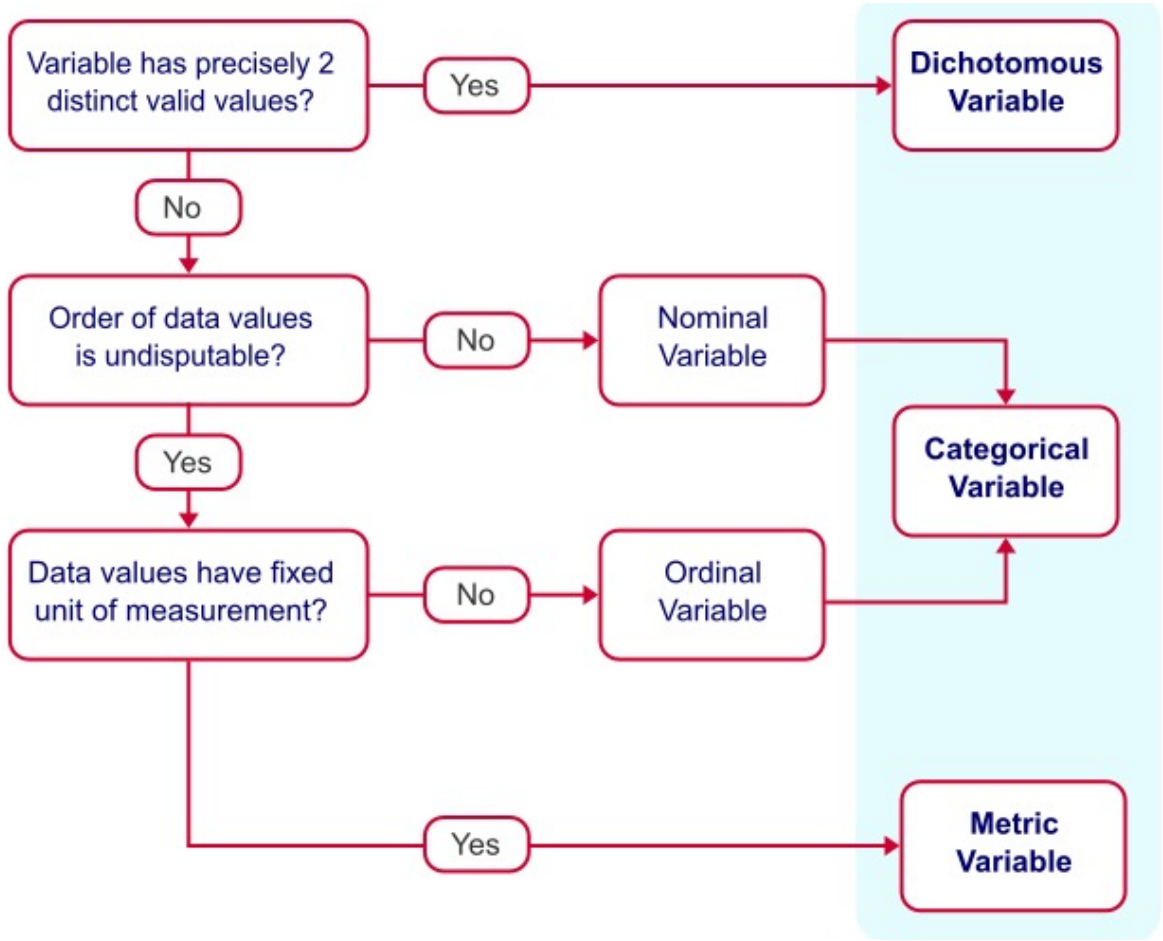
LEVELS OF DATA: LEVELS OF MEASUREMENTS/OBSERVATIONS

SCALE		EXAMPLE
Nominal		Gender
Ordinal		Position in race
Interval		Temperature (in Fahrenheit)
Ratio		Money



LEVELS OF DATA: What data to measure or observe?





LEVELS OF DATA: What data to measure or observe?

Differences between measurements, true zero exists

Ratio Data

Differences between measurements but no true zero

Interval Data

Quantitative Data

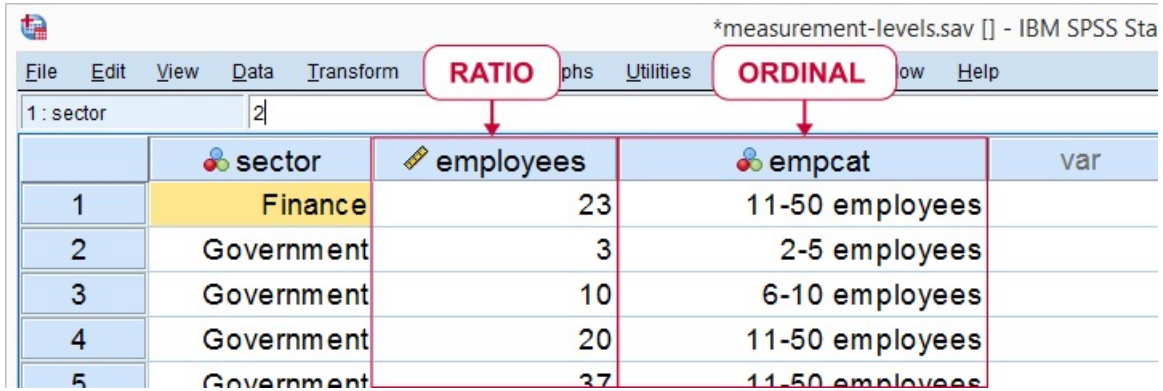
Ordered Categories (rankings, order, or scaling)

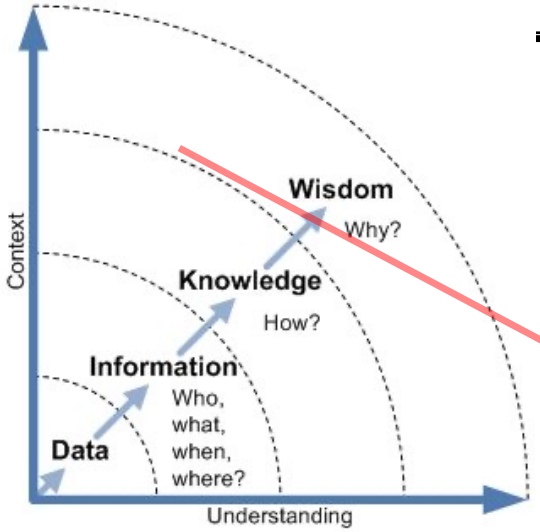
Ordinal Data

Categories (no ordering or direction)

Nominal Data

Qualitative Data





Context Independent

Wisdom is the recognition that knowledge patterns arise from fundamental principles and the understanding of what those principles are.

Wisdom

UNDERSTANDING PRINCIPLES

Knowledge is represented by patterns among data, information and possibly other knowledge. These patterns don't actually constitute knowledge until they are understood.

Knowledge

UNDERSTANDING PATTERNS

Information represented by relationships between data and possibly other information

Information

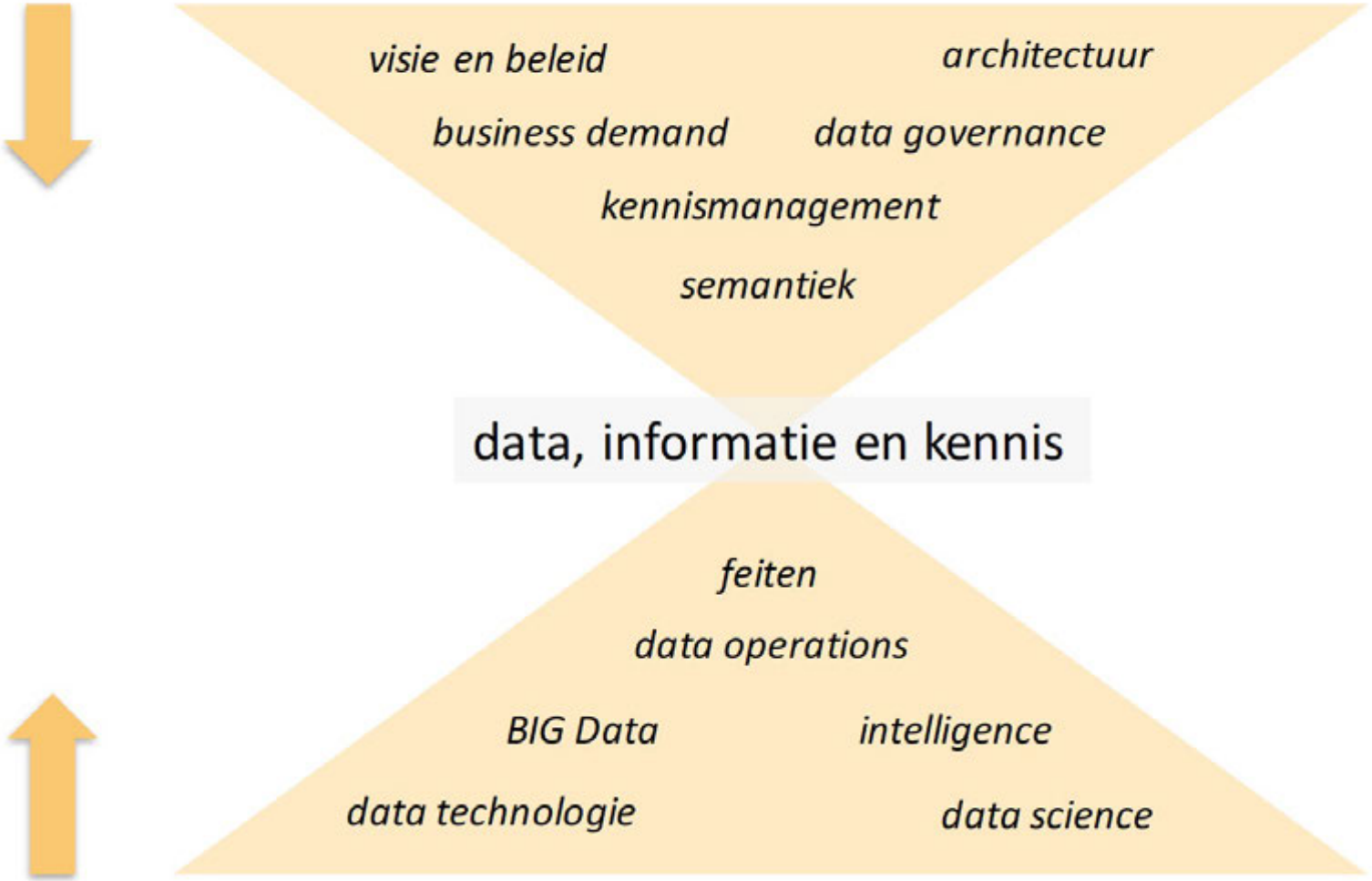
UNDERSTANDING RELATIONS

Data is an item or event out of context with no relation to Other things.

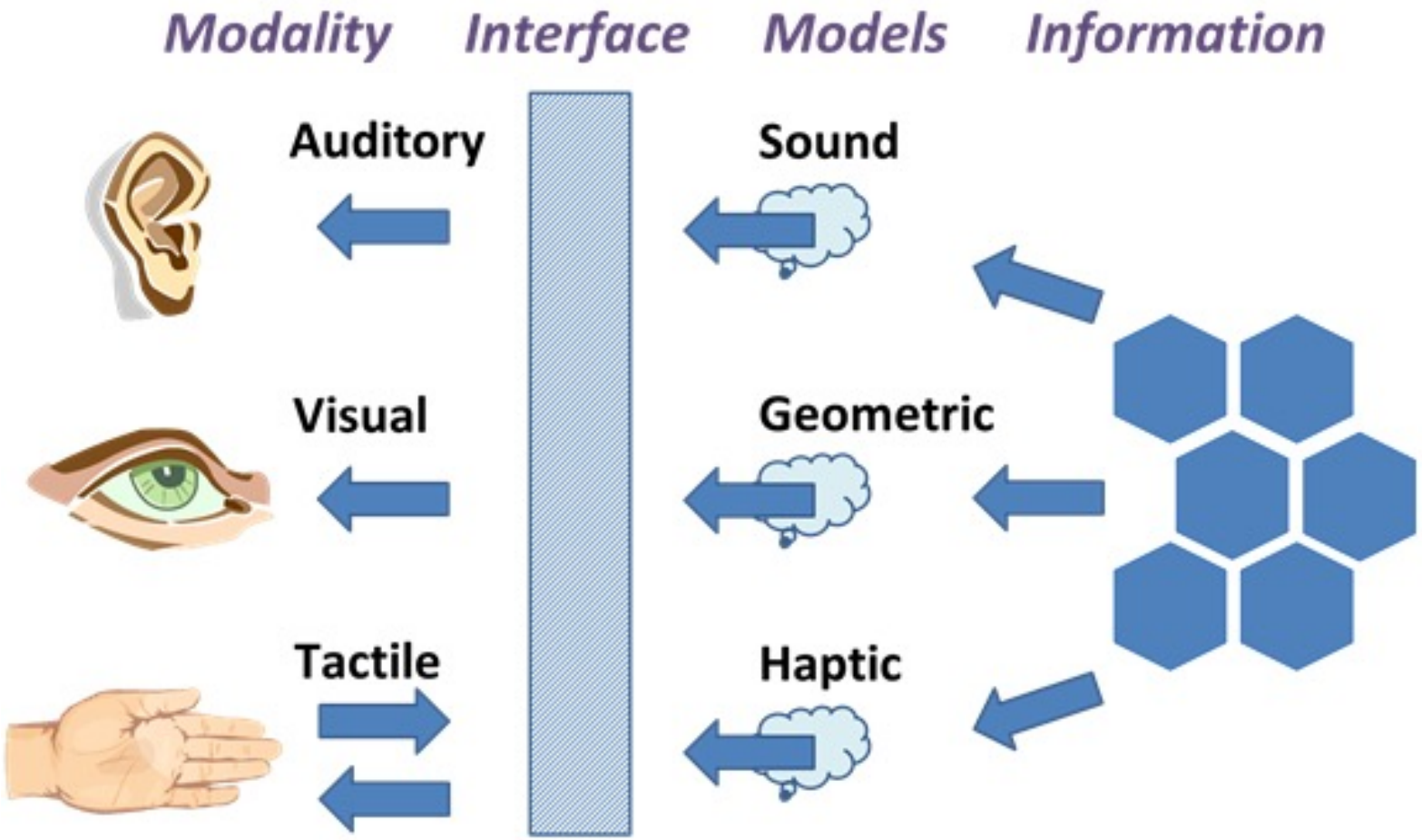
Data

Understanding

Data, informatie en kennis is wat ons verbindt



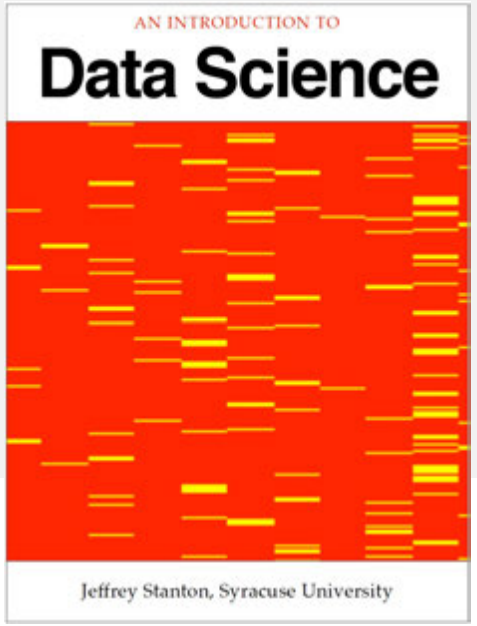
http://intelligence.agconnect.nl/content/van-data-naar-informatie


Modality
Interface
Models
Information
Auditory
Visual
Tactile
Sound
Geometric
Haptic

DATA-DRIVEN: Data versus Information

Data [gegevens]

Raw Facts
No Context
Numbers
Symbols

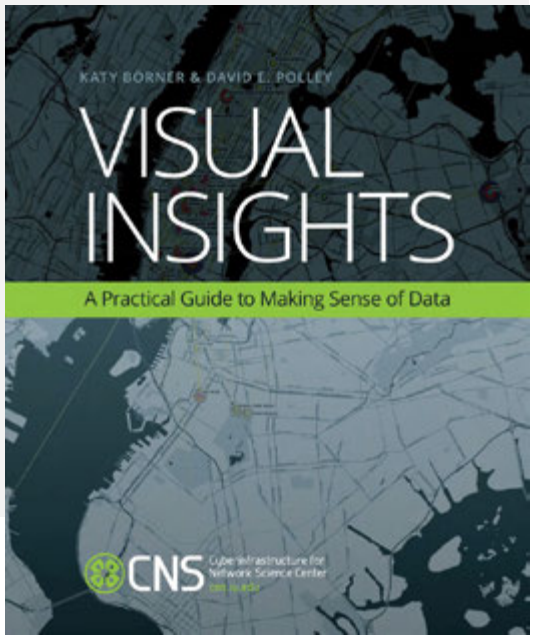


Information

Data with structure = processed data

Value-added to Data through

- Summarisation
- Organisation
- Analysing



TYPES OF DATA: Structured vs Unstructured [organized vs unorganized]

Data Structuring

Structured (organized) data: This is data that can be thought of as observations and characteristics. It is usually organized using a table method (rows and columns).

Unstructured (unorganized) data: This data exists as a free entity and does not follow any standard organization hierarchy.

Structured Data
High Degree of organization, such as a relational database

Column	Value
Patient	Joe Brown
Date of Birth	02/13/1972
Date Admitted	02/05/2014

Unstructured Data
Information that is difficult to organize using traditional mechanisms

"The patient came in complaining of chest pain, shortness of breath, and lingering headaches…smokes 2 packs a day… family history of heart disease…has been experiencing similar symptoms for the past 12 hours…."

DATA STRUCTURING: Generalized Form of a Data Table

Data Table [DATA MATRIX]

A generalized version of the data table is shown.

This table can represent any number of observations described over multiple variables.

This table describes a series of observations (from o1 to on) where each observation is described using a series of variables (from x1 to xp). A value is provided for each variable of each observation.

		Variables				
Observations		x_1	x_2	x_3	...	x_p
	o_1	x_{11}	x_{12}	x_{13}	...	x_{1p}
	o_2	x_{21}	x_{22}	x_{23}	...	x_{2p}
	o_3	x_{31}	x_{32}	x_{33}	...	x_{3p}
	...	...	...	...	...	...
	o_n	x_{n1}	x_{n2}	x_{n3}	...	x_{np}

Most data that exists in text form, including server logs and Facebook posts, is unstructured

Scientific observations, as recorded by careful scientists, are kept in a very neat and organized (structured) format: THE DATA TABLE

A genetic sequence of chemical nucleotides [ACGTATTGCA] is unstructured even if the order of the nucleotides matters

DATA STRUCTURING: Observations versus Variables

Data Table [DATA MATRIX]

A generalized version of the data table is shown. This table can represent any number of observations described over multiple variables.

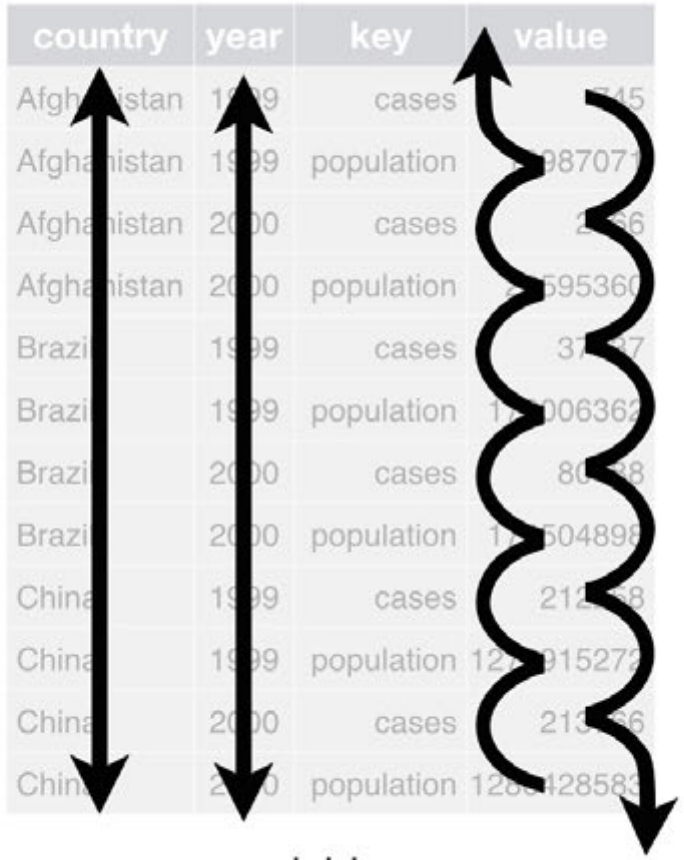
This table describes a series of observations (from o1 to on) where each observation is described using a series of variables (from x1 to xp). A value is provided for each variable of each observation.

		Variables				
Observations		x_1	x_2	x_3	...	x_p
	o_1	x_{11}	x_{12}	x_{13}	...	x_{1p}
	o_2	x_{21}	x_{22}	x_{23}	...	x_{2p}
	o_3	x_{31}	x_{32}	x_{33}	...	x_{3p}
	...	...	...	...	...	...
	o_n	x_{n1}	x_{n2}	x_{n3}	...	x_{np}

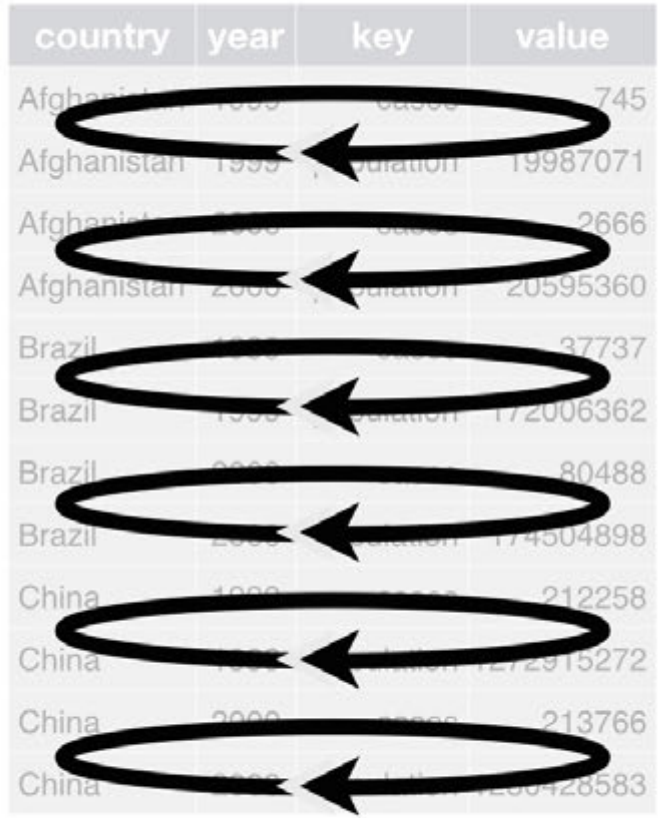
Patient ID	Treated	Age	Outcome	Random
1	Yes	Young	Positive	0.24
2	No	Young	Positive	0.85
3	Yes	Old	Negative	0.64
4	No	Old	Negative	0.70
5	No	Old	Negative	0.87
6	No	Old	Negative	0.72
7	No	Old	Negative	0.86
8	No	Young	Negative	0.16
9	No	Young	Positive	0.17

country	year	key	value
Afghanistan	1999	cases	745
Afghanistan	1999	population	19987071
Afghanistan	2000	cases	2666
Afghanistan	2000	population	20595360
Brazil	1999	cases	37737
Brazil	1999	population	172006362
Brazil	2000	cases	80488
Brazil	2000	population	174504898
China	1999	cases	212258
China	1999	population	1272915272
China	2000	cases	213766
China	2000	population	1280428583

table2



variables



observations

STRUCTURED DATA FORMATS

- CSV, comma separated values

```
sepal_length,sepal_width,petal_length,petal_width,species
5.1,3.5,1.4,0.2,setosa
4.9,3,1.4,0.2,setosa
4.7,3.2,1.3,0.2,setosa
4.6,3.1,1.5,0.2,setosa
```

- hierarchies, e.g., Newick tree format

```
(A,B,(C,D)E)F;
```

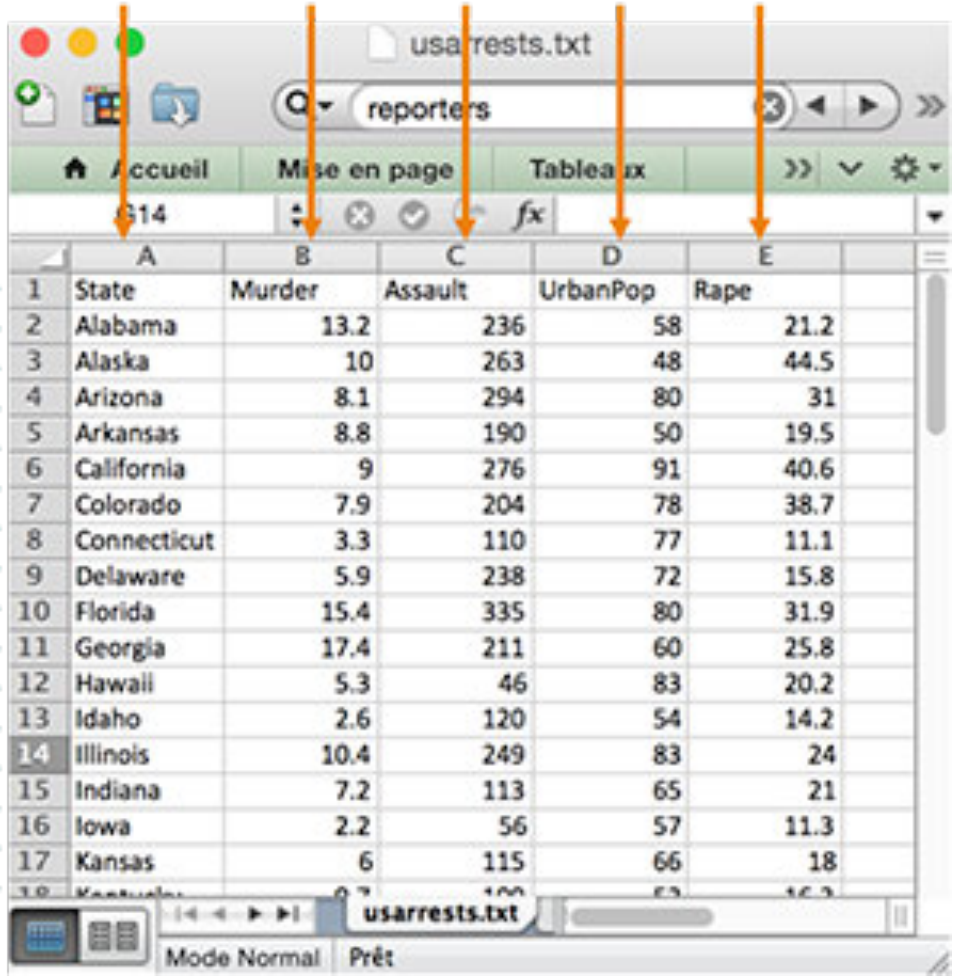
- networks, e.g., GraphViz (DOT)

```
digraph graphname {
    a -> b -> c;
    b -> d;
}
```

Tidy data

Variables

Observations



usarrests.txt

	A	B	C	D	E
1	State	Murder	Assault	UrbanPop	Rape
2	Alabama	13.2	236	58	21.2
3	Alaska	10	263	48	44.5
4	Arizona	8.1	294	80	31
5	Arkansas	8.8	190	50	19.5
6	California	9	276	91	40.6
7	Colorado	7.9	204	78	38.7
8	Connecticut	3.3	110	77	11.1
9	Delaware	5.9	238	72	15.8
10	Florida	15.4	335	80	31.9
11	Georgia	17.4	211	60	25.8
12	Hawaii	5.3	46	83	20.2
13	Idaho	2.6	120	54	14.2
14	Illinois	10.4	249	83	24
15	Indiana	7.2	113	65	21
16	Iowa	2.2	56	57	11.3
17	Kansas	6	115	66	18

KINDS OF DATA

- STRUCTURED DATA
  - lists
  - $n \times p$ tables, arrays
  - hierarchies (e.g., organization chart)
  - networks (e.g., travel routes, hypertext = links)
- Generic data-interchange formats: XML, JSON

- UNSTRUCTURED DATA
  - text
  - images
  - video
  - sound
- Often can be made structured by, e.g., parsing language, segmenting images, etc.

BIG DATA

- A crucial part of the rise of Data Science is the steep increase in the amount and availability of data

- Big Data refers not only to the quantity but also to the quality of the data:
  - VOLUME: lots of it
  - VELOCITY: fast (streaming)
  - VARIETY: all kinds, not nice and "clean"
  - VERACITY: can it be trusted?

JSON

- Similar to XML but simpler

```
{
  "firstName": "John",
  "lastName": "Smith",
  "isAlive": true,
  "age": 25,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021-3100"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "office",
      "number": "646 555-4567"
    },
    {
      "type": "mobile",
      "number": "123 456-7890"
    }
  ],
  "children": [],
  "spouse": null
}
```

HOGESCHOOL ROTTERDAM

XML

- same example:

```
<person>
  <firstName>John</firstName>
  <lastName>Smith</lastName>
  <age>25</age>
  <address>
    <streetAddress>21 2nd Street</streetAddress>
    <city>New York</city>
    <state>NY</state>
    <postalCode>10021</postalCode>
  </address>
  <phoneNumber>
    <type>home</type>
    <number>212 555-1234</number>
  </phoneNumber>
  <phoneNumber>
    <type>fax</type>
    <number>646 555-4567</number>
  </phoneNumber>
  <gender>
    <type>male</type>
  </gender>
</person>
```

PARSING

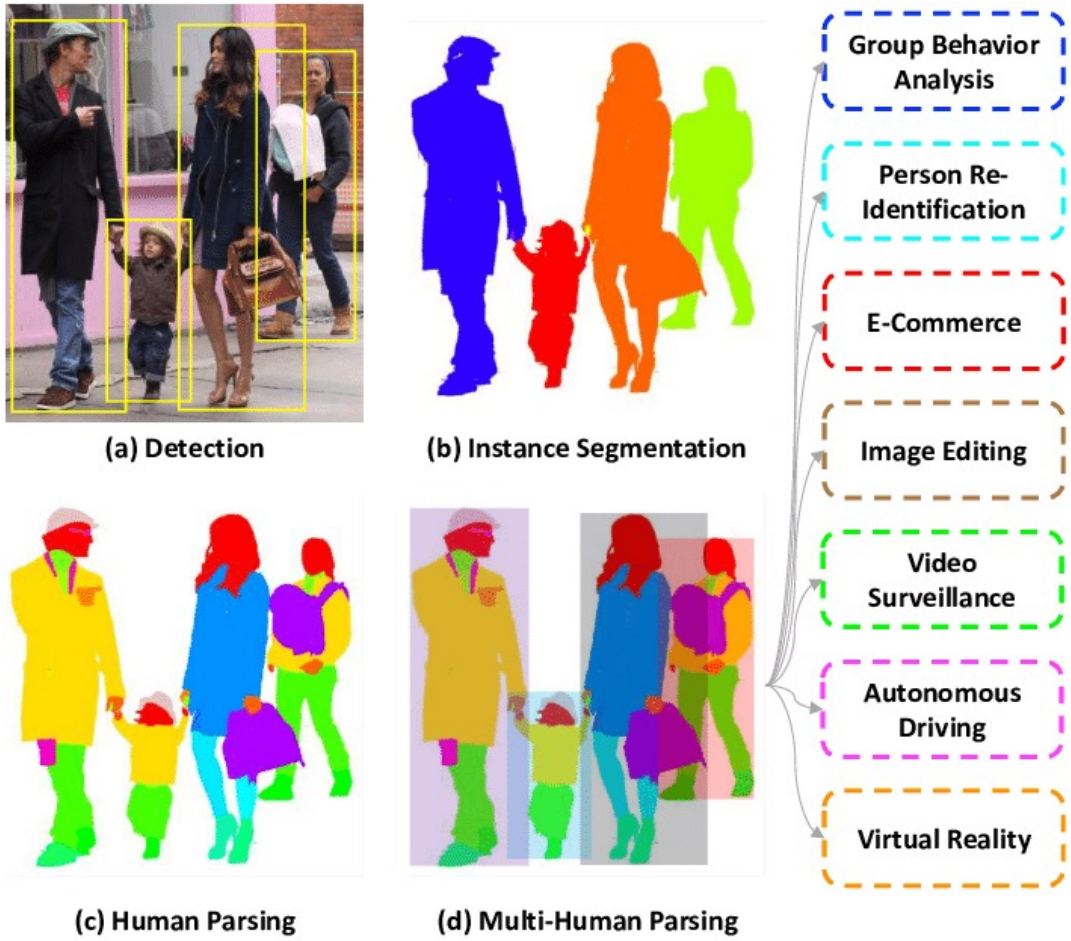
- Given a known grammar, unstructured text data can be parsed
- “It ain’t over till the fat lady sings”

  ((it, (ain't, over)), (till, ((the, (fat, lady)), sings)))

- Similarly, images can be segmented into parts

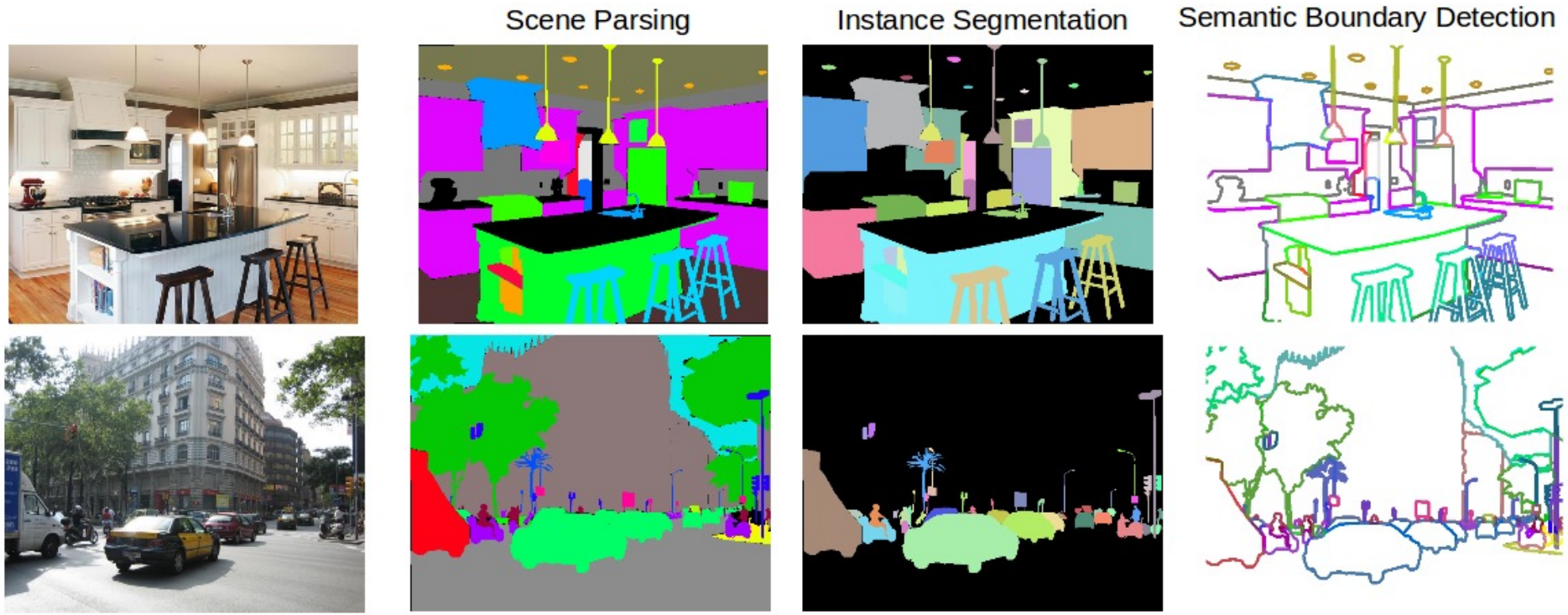
PARSING

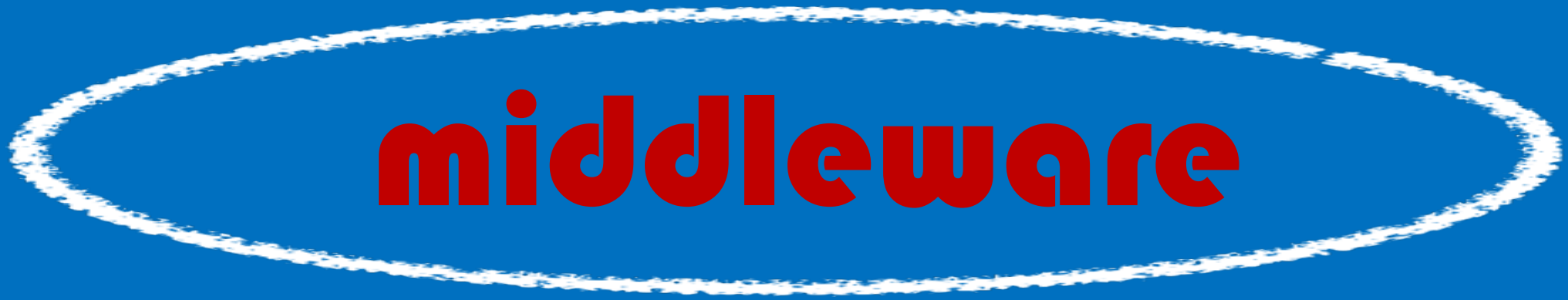
- Similarly, images can be segmented into parts



PARSING

- Similarly, images can be segmented into parts

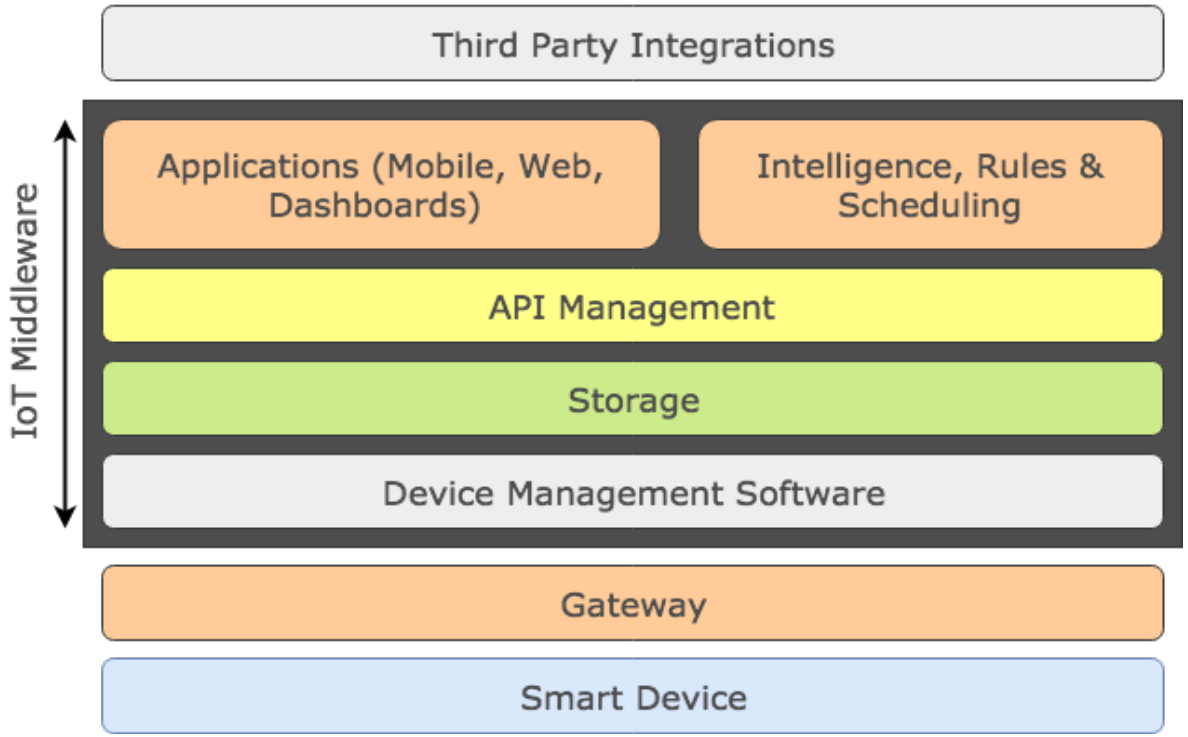




HOGESCHOOL ROTTERDAM

Getting Started with (the) IoT

IoT Platforms == MiddleWare



IoT Platforms / MiddleWare

Why do we need these platforms? Why can't we build one of our own?

We can; however, first we have to consider the following points:

- How long does it take for one to build a piece of end-to-end, bug free IoT middleware?
- Your resources' time versus money spent on building this middleware.
- Your in-house team's ability to build an IoT middleware.
- How generic can you build it? Will this IoT middleware scale for all types of applications?

Since almost all IoT platforms have a similar set of features, using a continuously improving, community contributed platform is always better than building one on your own.

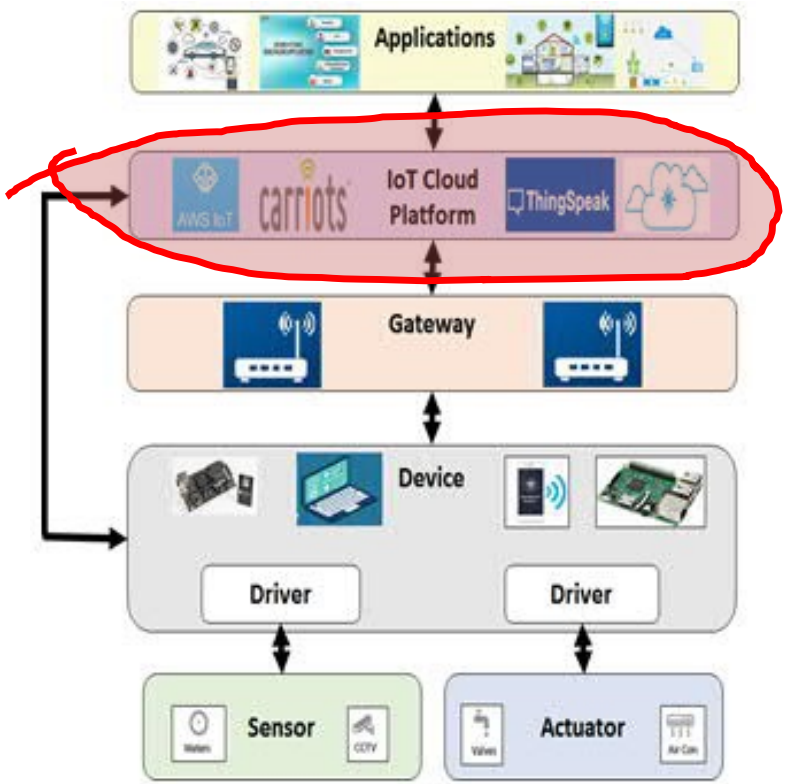
IoT platforms

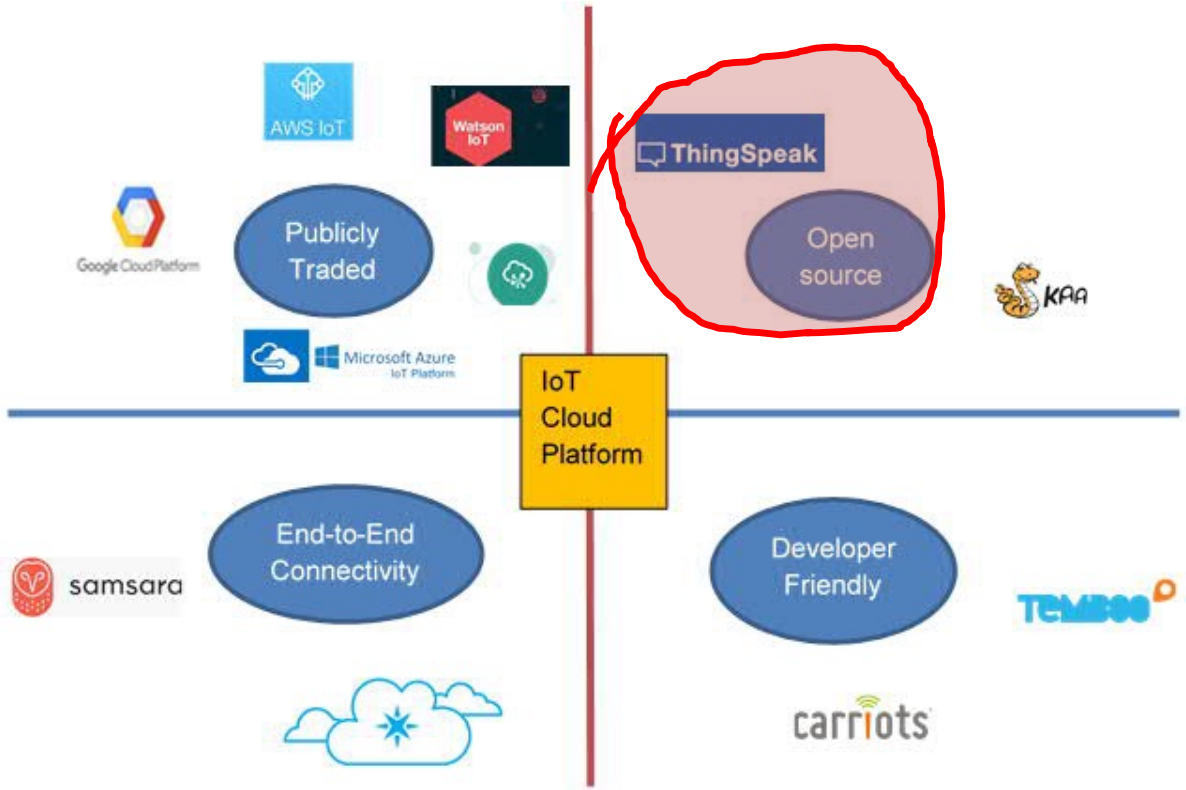
Now that we have a sense of why we are using an existing IoT platform for our IoT solutions, let us look at the options we have for these platforms. The following are a few popular IoT platforms.

You can visit the site links next to the provider to read more about the offerings:

- **AWS IoT**: https://aws.amazon.com/iot/
- **Microsoft Azure IoT**: https://www.microsoft.com/en-in/internet-of-things/azure-iot-suite
- **IBM Watson**: https://www.ibm.com/internet-of-things
- **Google Cloud IoT**: https://cloud.google.com/solutions/iot/
- **Cisco IoT Cloud Connect**: https://www.jasper.com/
- **Salesforce IoT cloud**: https://www.salesforce.com/eu/iot-cloud/
- **Bosch IoT Suite**: https://www.bosch-si.com/iot-platform/bosch-iot-suite/homepage-bosch-iot-suite.html
- **Kaa IoT**: https://www.kaaproject.org/
- **ThingSpeak**: https://thingspeak.com/
- **DeviceHive**: https://devicehive.com/

IoT Platforms / MiddleWare





http://www.thetips4you.com/iot-best-open-source-applications-open-source-industrial-iot-platform

Setting Up Your IoT Device

The Broker. The broker acts as a gateway; it receives messages from a publisher (a client) and delivers the messages to a subscriber (another client). Brokers are sometimes referred to as *servers*.

The Subscriber. The subscriber declares its topics of interest to the broker, and the broker sends messages published to those topics.

The Publisher. The publisher sends messages to the broker using a name- space or a topic name, and the broker forwards the messages to the respective subscribers.

ThingSpeak™ Channels Apps Support Commercial Use How to Buy Sign In Sign Up

Learn More About ThingSpeak

ThingSpeak™ is an IoT analytics platform service that allows you to aggregate, visualize and analyze live data streams in the cloud. ThingSpeak provides instant visualizations of data posted by your devices to ThingSpeak. With the ability to execute MATLAB® code in ThingSpeak you can perform online analysis and processing of the data as it comes in. ThingSpeak is often used for prototyping and proof of concept IoT systems that require analytics.

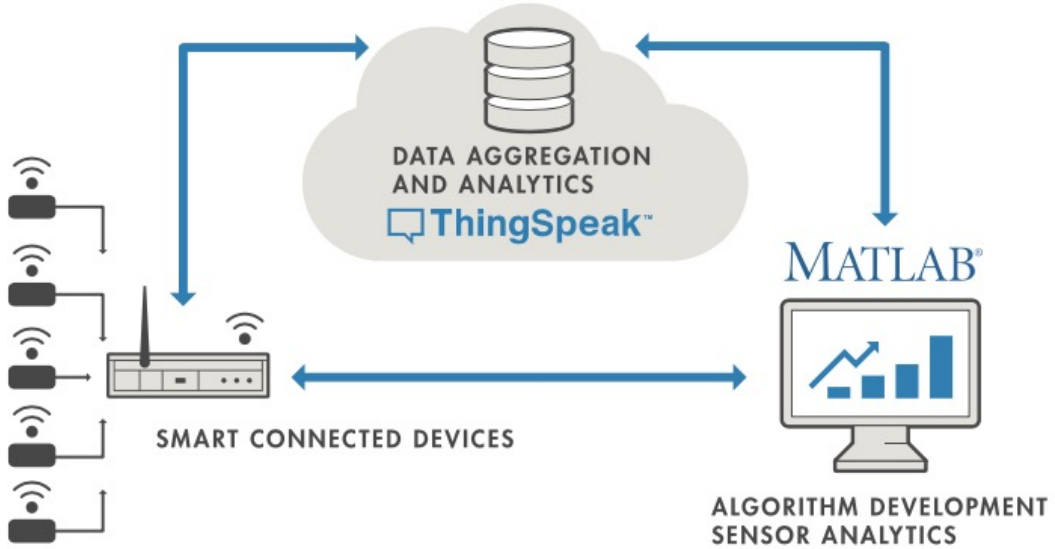
What is IoT?

Internet of Things (IoT) describes an emerging trend where a large number of embedded devices (things) are connected to the Internet. These connected devices communicate with people and other things and often provide sensor data to cloud storage and cloud computing resources where the data is processed and analyzed to gain important insights. Cheap cloud computing power and increased device connectivity is enabling this trend.

IoT solutions are built for many vertical applications such as environmental monitoring and control, health monitoring, vehicle fleet monitoring, industrial monitoring and control, and home automation.

At a high level, many IoT systems can be described using the diagram below:

https://thingspeak.com/pages/learn_more



ThingSpeak™ Channels Apps Support Commercial Use How to Buy Account Sign Out

Bulk-Update Using a Raspberry Pi Board

Channel ID: 920549
Author: robfvdw
Access: Public

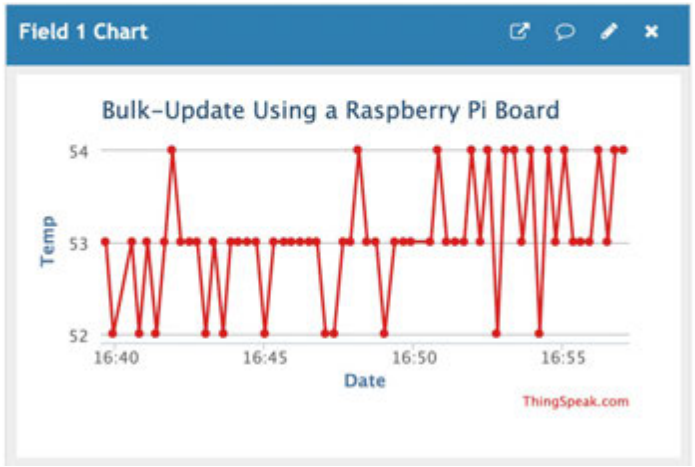
This example shows how to use a Raspberry Pi™ board that runs Python® 2.7 that is connected to a Wi-Fi® network to collect data. You can continuously collect CPU temperature and percentage of CPU utilization over 15 seconds and bulk-update a ThingSpeak™ c

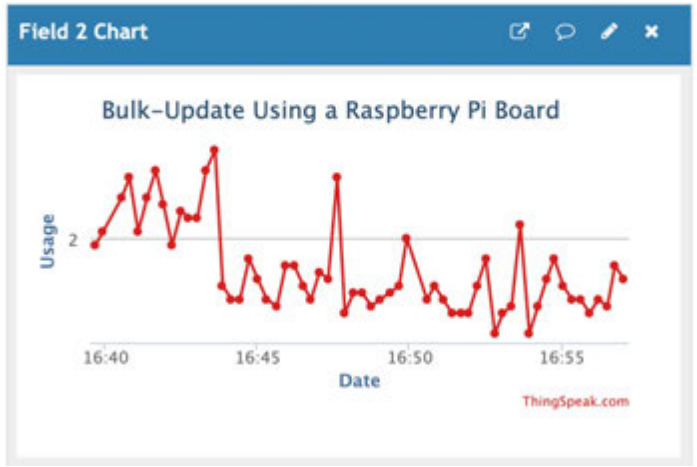
Private View | Public View | Channel Settings | Sharing | API Keys | Data Import / Export

Add Visualizations | Add Widgets | Export recent data | MATLAB Analysis | MATLAB Visualization

Channel Stats

Created: about a month ago
Last entry: less than a minute ago
Entries: 4816





To better understand the ThingSpeak platform, we will build a sample end-to-end IoT application.

Performed will be the following tasks:

- Send data to ThingSpeak via a Channel
- Integrate ThingSpeak with Raspberry Pi implemented via HTTP (LAN network)
- Visualize the sensor data

+ {Set thresholds on the data and receive alerts}

Bulk-Update Using a Raspberry Pi Board

HOGESCHOOL ROTTERDAM

You will learn how to Send sensor data to ThingSpeak via your own Public Channel

Shown is how to use a Raspberry Pi board that runs Python 2.7 that is connected to a Wi-Fi network to collect data.

You can continuously collect CPU temperature and percentage of CPU utilization over 15 seconds and bulk-update a ThingSpeak channel every 2 minutes.

Used is a Bulk-Write JSON Data API to collect sensor data as a batch and send it to your public ThingSpeak channels.

This bulk-update reduces the power usage of your devices.

Since the Raspberry Pi board does not come with a real-time clock, you can use the relative time stamp for bulk-update messages.

How to create a ThingSpeak IoT Cannel

Get Started
Learn the basics of ThingSpeak

Configure Accounts and Channels
Information on ThingSpeak channels, users, and licenses

Write Data to Channel
Use the REST and MQTT APIs to update channels with software or devices

Read Data from Channel
Use the REST and MQTT APIs to read channels using software or devices

Prepare and Analyze Data
Filter, transform, and respond to data in MATLAB

Visualize Data
Transform and visualize data in MATLAB

Act on Data
Use ThingSpeak apps to trigger an action or transform and visualize data

Specialized Analysis with MATLAB
ThingSpeak examples that show use of the advanced tools available in add-on toolboxes

API Reference
Use the REST and MQTT APIs to update ThingSpeak channels and to chart numeric data stored in channel

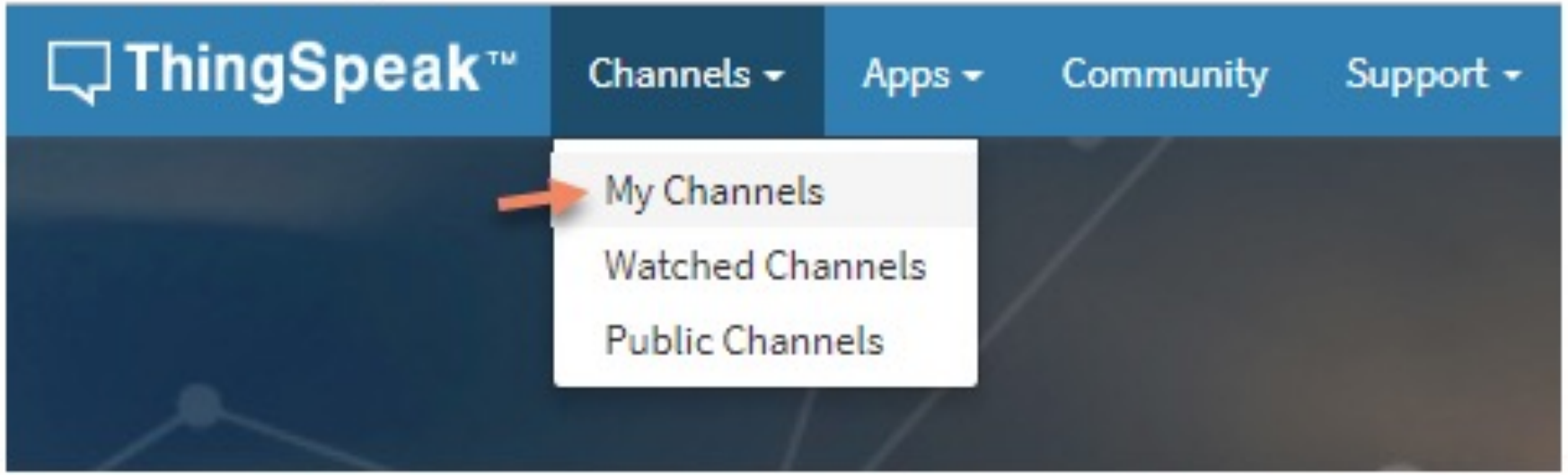
https://nl.mathworks.com/help/thingspeak/index.html?s_tid=CRUX_lftnav

Creating an TingSpeak account

If you do not have an account already, you can visit https://thingspeak.com/users/sign_up to create one.
Once you have created the account, activate it, and log in, you will see an interface like this:

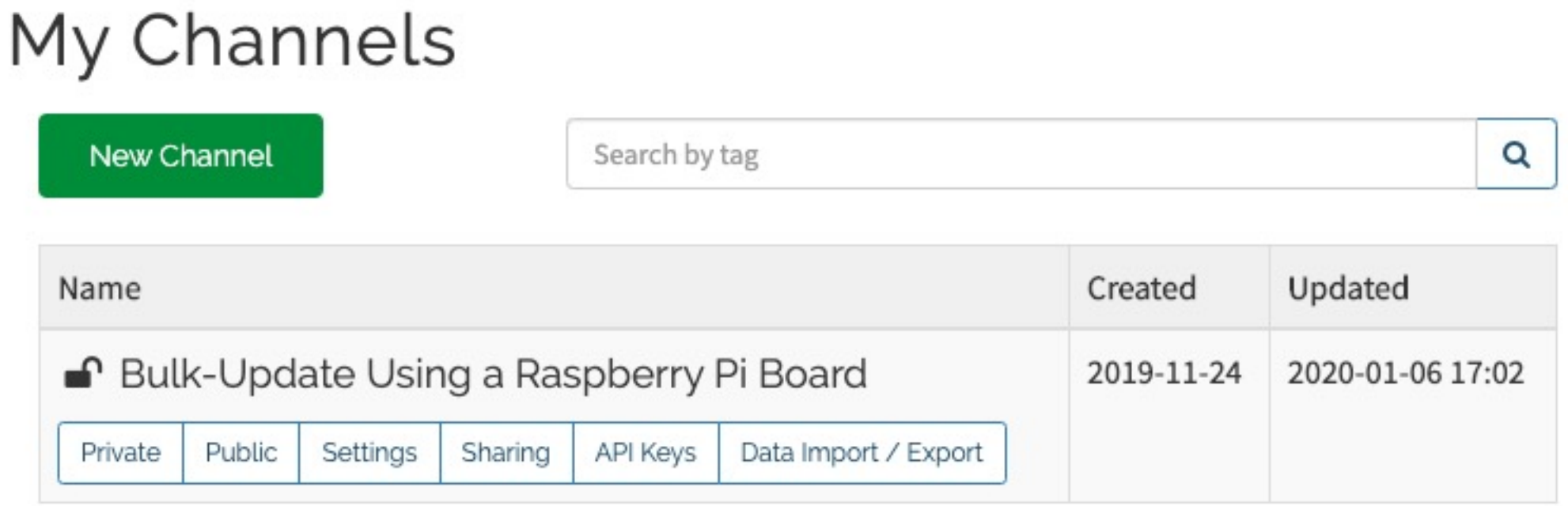
Create a Channel

1. Sign In to ThingSpeak™ using your MathWorks® Account, or create a new MathWorks account.
2. Click **Channels** > **MyChannels**.



Collect Data in a Public Channel called

“Bulk-Update Using a Raspberry Pi Board”



https://www.mathworks.com/help/thingspeak/collect-data-in-a-new-channel.html

Bulk-Update Using a Raspberry Pi Board

Channel ID: 920549
Author: robfvdw
Access: Public

This example shows how to use a Raspberry Pi™ board that runs Python® 2.7 that is connected to a Wi-Fi® network to collect data. You can continuously collect CPU temperature and percentage of CPU utilization over 15 seconds and bulk-update a ThingSpeak™ c

https://nl.mathworks.com/help/thingspeak/continuously-collect-data-and-bulk-update-a-thingspeak-channel-using-a-raspberry-pi-board.html

Private View | Public View | Channel Settings | Sharing | API Keys | Data Import / Export

Channel Settings

Percentage complete	50%
Channel ID	920549
Name	Bulk-Update Using a Raspberry Pi Board
Description	This example shows how to use a Raspberry Pi™ board that runs Python® 2.7 that is connected to a
Field 1	Temp ☑
Field 2	Usage ☑
Field 3	☐

Help

Channels store all the data that a ThingSpeak application collects. Each channel includes eight fields that can hold any type of data, plus three fields for location data and one for status data. Once you collect data in a channel, you can use ThingSpeak apps to analyze and visualize it.

Channel Settings

- Percentage complete: Calculated based on data entered into the various fields of a channel. Enter the name, description, location, URL, video, and tags to complete your channel.
- Channel Name: Enter a unique name for the ThingSpeak channel.
- Description: Enter a description of the ThingSpeak channel.
- Field#: Check the box to enable the field, and enter a field name. Each ThingSpeak channel can have up to 8 fields.
- Metadata: Enter information about channel data, including JSON, XML, or CSV data.

Bulk-Update Using a Raspberry Pi Board

Channel ID: 920549
Author: robfvdw
Access: Public

This example shows how to use a Raspberry Pi™ board that runs Python® 2.7 that is connected to a Wi-Fi® network to collect data. You can continuously collect CPU temperature and percentage of CPU utilization over 15 seconds and bulk-update a ThingSpeak™ c

Private View | Public View | Channel Settings | Sharing | API Keys | Data Import / Export

Write API Key

Key: X200JB02Z2LN5W8E

Generate New Write API Key

Read API Keys

Key: 2CBID06MIN1M06KZ

Note:

Save Note | Delete API Key

Add New Read API Key

Help

API keys enable you to write data to a channel or read data from a private channel. API keys are auto-generated when you create a new channel.

API Keys Settings

- Write API Key: Use this key to write data to a channel. If you feel your key has been compromised, click Generate New Write API Key.
- Read API Keys: Use this key to allow other people to view your private channel feeds and charts. Click Generate New Read API Key to generate an additional read key for the channel.
- Note: Use this field to enter information about channel read keys. For example, add notes to keep track of users with access to your channel.

API Requests

Write a Channel Feed

```
GET https://api.thingspeak.com/update?api_key=X200JB02Z2LN5W8E
```

Read a Channel Feed

```
GET https://api.thingspeak.com/channels/920549/feeds.json?resu
```

Read a Channel Field

```
GET https://api.thingspeak.com/channels/920549/fields/1.json?r
```

Read Channel Status Updates

```
GET https://api.thingspeak.com/channels/920549/status.json
```

Bulk-Update Using a Raspberry Pi Board

Watch | Like 0 | Share

Channel ID: 920549
Author: robfvdw
Access: Public

This example shows how to use a Raspberry Pi™ board that runs Python® 2.7 that is connected to a Wi-Fi® network to collect data. You can continuously collect CPU temperature and percentage of CPU utilization over 15 seconds and bulk-update a ThingSpeak™ c

Private View | Public View | Channel Settings | Sharing | API Keys | Data Import / Export

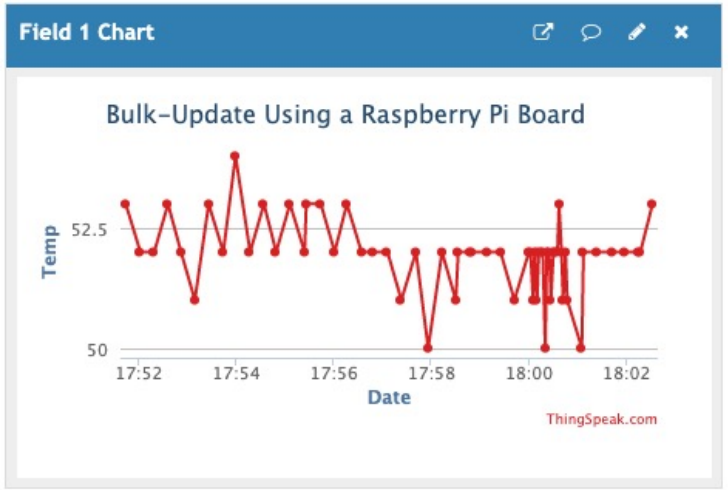
Add Visualizations | Add Widgets | Export recent data

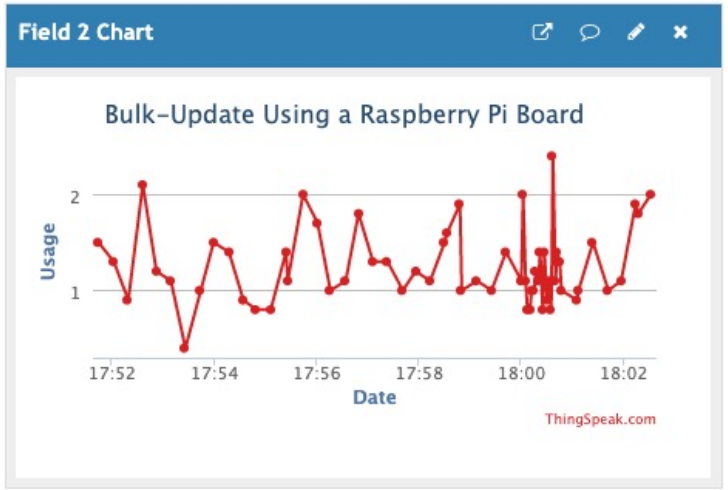
MATLAB Analysis | MATLAB Visualization

Channel Stats

Created: about a month ago
Last entry: about 2 hours ago
Entries: 5150

https://thingspeak.com/channels/920549/





Writing Python code to collect data in a public channel called "Bulk-Update Using a Raspberry Pi Board"

```
#1) Create a channel as shown in Collect Data in a Public Channel called "Bulk-Update Using a Raspberry Pi Board"

#2) Import the necessary libraries for the script.
import json
import time
import os
import psutil
import urllib2 as ul

#3) Define global variables that track the last connection time and last update time. Also, define time intervals to update the data, and post the data to ThingSpeak.
lastConnectionTime = time.time() # Track the last connection time
lastUpdateTime = time.time() # Track the last update time
postingInterval = 120 # Post data once every 2 minutes
updateInterval = 15 # Update once every 15 seconds
```

```
#4) Define your ThingSpeak channel settings such as write API key and channel ID along with ThingSpeak server settings.
writeAPIkey = "YOUR-CHANNEL-WRITEAPIKEY" # Replace YOUR-CHANNEL-WRITEAPIKEY with your channel write API key
channelID = "YOUR-CHANNELID" # Replace YOUR-CHANNELID with your channel ID
url = "https://api.thingspeak.com/channels/"+channelID+"/bulk_update.json" # ThingSpeak server settings
messageBuffer = []
```

```
writeAPIkey = "X2O0JB02Z2LN5W8E" # Replace YOUR-CHANNEL-WRITEAPIKEY with your channel write API key
channelID = "920549" # Repla                    LID with your channel ID
url = "https://api.thingspeak.com/channels/"+channelID+"/bulk_update.json" # ThingSpeak server settings
messageBuffer = []
```

Explorer (⇧⌘E)

Writing Python code to collect data in a public channel called "Bulk-Update Using a Raspberry Pi Board"

```
#5) Define the function httpRequest to send data to ThingSpeak and to print the response code from the server. A response code 202 indicates tha
def httpRequest():
    '''Function to send the POST request to
    ThingSpeak channel for bulk update.'''

    global messageBuffer
    data = json.dumps({'write_api_key':writeAPIkey,'updates':messageBuffer}) # Format the json data buffer
    req = ul.Request(url = url)
    requestHeaders = {"User-Agent":"mw.doc.bulk-update (Raspberry Pi)","Content-Type":"application/json","Content-Length":str(len(data))}
    for key,val in requestHeaders.iteritems(): # Set the headers
        req.add_header(key,val)
    req.add_data(data) # Add the data to the request
    # Make the request to ThingSpeak
    try:
        response = ul.urlopen(req) # Make the request
        print response.getcode() # A 202 indicates that the server has accepted the request
    except ul.HTTPError as e:
        print e.code # Print the error code
    messageBuffer = [] # Reinitialize the message buffer
    global lastConnectionTime
    lastConnectionTime = time.time() # Update the connection time
```

Writing Python code to collect data in a public channel called "Bulk-Update Using a Raspberry Pi Board"

```
#6) Define the function getData that returns the CPU temperature in Celsius along with the CPU utilization as a percentage.def getData():
'''Function that returns the CPU temperature and percentage of CPU utilization'''
    cmd = '/opt/vc/bin/vcgencmd measure_temp'
    process = os.popen(cmd).readline().strip()
    cpuTemp = process.split('=')[1].split("'")[0]
    cpuUsage = psutil.cpu_percent(interval=2)
    return cpuTemp,cpuUsage
```

Wrting Python code to collect data in a public channel called "Bulk-Update Using a Raspberry Pi Board"

```
#7) Define the function updatesJson to continuously update the message buffer every 15 seconds.
def updatesJson():
    '''Function to update the message buffer
    every 15 seconds with data. And then call the httpRequest
    function every 2 minutes. This examples uses the relative timestamp as it uses the "delta_t" parameter.
    If your device has a real-time clock, you can also provide the absolute timestamp using the "created_at" parameter.
    '''

    global lastUpdateTime
    message = {}
    message['delta_t'] = int(round(time.time() - lastUpdateTime))
    Temp,Usage = getData()
    message['field1'] = Temp
    message['field2'] = Usage
    global messageBuffer
    messageBuffer.append(message)
    # If posting interval time has crossed 2 minutes update the ThingSpeak channel with your data
    if time.time() - lastConnectionTime >= postingInterval:
        httpRequest()
    lastUpdateTime = time.time()
```

Writing Python code to collect data in a public channel called "Bulk-Update Using a Raspberry Pi Board"

```
#8) Run an infinite loop to continuously call the function updatesJson once every 15 seconds.

if __name__ == "__main__":  # To ensure that this is run directly and does not run when imported
    while 1:
        # If update interval time has crossed 15 seconds update the message buffer with data
        if time.time() - lastUpdateTime >= updateInterval:
            updatesJson()
```

```python
import json
import time
import os
import psutil
import urllib2 as ul

lastConnectionTime = time.time() # Track the last connection time
lastUpdateTime = time.time() # Track the last update time
postingInterval = 120 # Post data once every 2 minutes
updateInterval = 15 # Update once every 15 seconds

writeAPIkey = "X2O0JB02Z2LN5W8E" # Replace YOUR-CHANNEL-WRITEAPIKEY with your channel write API key
channelID = "920549" # Replace YOUR-CHANNELID with your channel ID
url = "https://api.thingspeak.com/channels/"+channelID+"/bulk_update.json" # ThingSpeak server settings
messageBuffer = []

def httpRequest():
    '''Function to send the POST request to
    ThingSpeak channel for bulk update.'''

    global messageBuffer
    data = json.dumps({'write_api_key':writeAPIkey,'updates':messageBuffer}) # Format the json data buffer
    req = ul.Request(url = url)
    requestHeaders = {"User-Agent":"mw.doc.bulk-update (Raspberry Pi)","Content-Type":"application/
    json","Content-Length":str(len(data))}
    for key,val in requestHeaders.iteritems(): # Set the headers
        req.add_header(key,val)
    req.add_data(data) # Add the data to the request
    # Make the request to ThingSpeak
    try:
        response = ul.urlopen(req) # Make the request
        print response.getcode() # A 202 indicates that the server has accepted the request
    except ul.HTTPError as e:
        print e.code # Print the error code
    messageBuffer = [] # Reinitialize the message buffer
    global lastConnectionTime
    lastConnectionTime = time.time() # Update the connection time

def getData():
    '''Function that returns the CPU temperature and percentage of CPU utilization'''
    cmd = '/opt/vc/bin/vcgencmd measure_temp'
    process = os.popen(cmd).readline().strip()
    cpuTemp = process.split('=')[1].split("'")[0]
    cpuUsage = psutil.cpu_percent(interval=2)
    return cpuTemp,cpuUsage
```

```python
def updatesJson():
    '''Function to update the message buffer
    every 15 seconds with data. And then call the httpRequest
    function every 2 minutes. This examples uses the relative timestamp as it uses the "delta_t" parameter.
    If your device has a real-time clock, you can also provide the absolute timestamp using the "created_at"
    parameter.
    '''

    global lastUpdateTime
    message = {}
    message['delta_t'] = int(round(time.time() - lastUpdateTime))
    Temp,Usage = getData()
    message['field1'] = Temp
    message['field2'] = Usage
    global messageBuffer
    messageBuffer.append(message)


# If posting interval time has crossed 2 minutes update the ThingSpeak channel with your data
if time.time() - lastConnectionTime >= postingInterval:
        httpRequest()
        lastUpdateTime = time.time()

if __name__ == "__main__":  # To ensure that this is run directly and does not run when imported
    while 1:
        # If update interval time has crossed 15 seconds update the message buffer with data
        if time.time() - lastUpdateTime >= updateInterval:
            updatesJson()

            httpRequest()
```

HOGESCHOOL ROTTERDAM

OPDRACHT WEEK 6:

- Create a repository entry in GitHub where you describe al the step necessary to create a Raspberry Pi based IoT project using ThinkSpeak as middleware

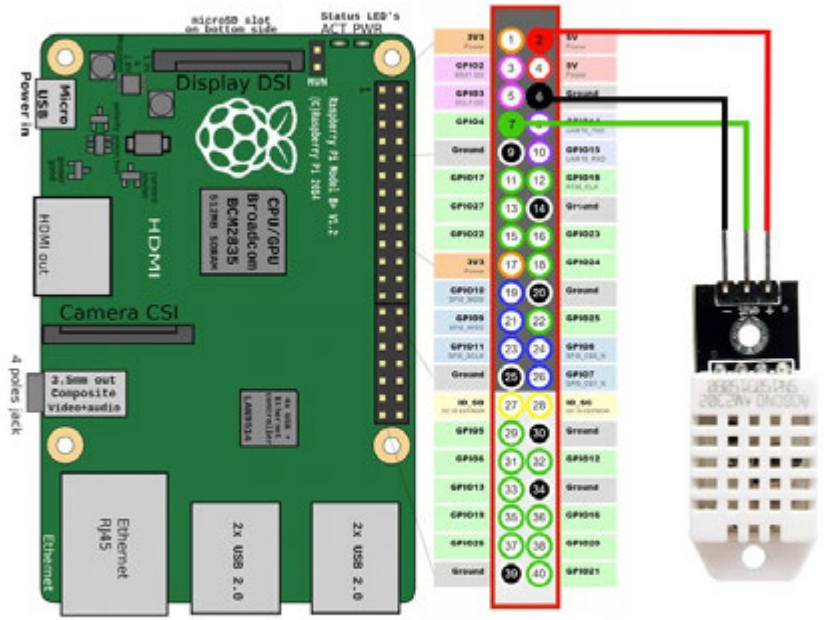
- Send {your sensor} data to ThingSpeak via your own public Channel as described in this tutorial, provide evidence on Git

Raspberry Pi Sensor to ThingsBoard using Constrained Application Protocol

HOGESCHOOL ROTTERDAM

Installation

Connecting the sensor



To connect the DHT22 sensor to the RPi:

1. Connect the `-` output on the sensor the pin 6 on the RPi.
2. Connect the `+` input on the sensor to pin 2 on the RPi.
3. Connect the `output` pin on the sensor to pin 7 on the RPi.

https://github.com/Silver292/rpi-coap

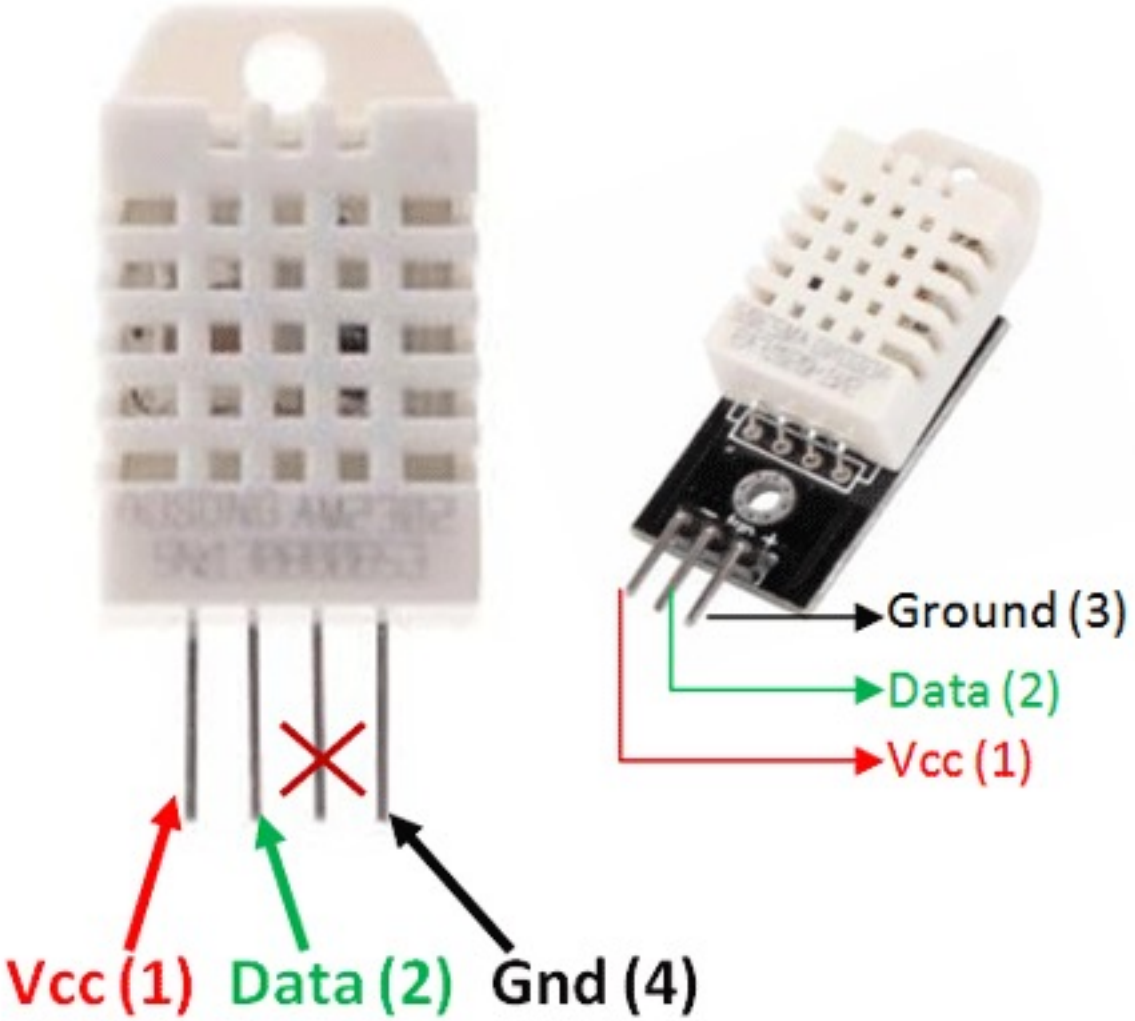
Introduction

This script is designed to be run on Raspberry Pi 3 hardware running the Raspbian 9.6 operating system.
It is also assumed that a DHT22 temperature and humidity sensor is connected to the Raspberry Pi (RPi) using GPIO 4 pin.
The process of connecting the sensor is described here.

The script creates a Constrained Application Protocol (CoAP) client on the RPi and repeatedly polls the attached DHT22 sensor for temperature and humidity data.

This data is then formatted to JavaScript Object Notation (JSON) and is sent to a CoAP endpoint provided by the ThingsBoard Cloud Platform.

This data is then displayed on the ThingsBoard dashboard that shows the current temperature and humidity as well as historical readings in a graph form.


Ground (3)
Data (2)
Vcc (1)
Vcc (1)
Data (2)
Gnd (4)

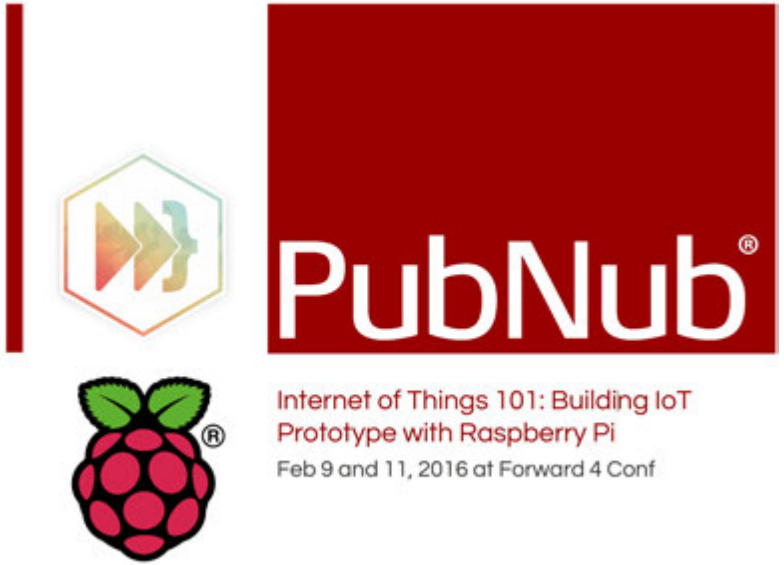
ThingSpeak Examples
Temp Sensor + Raspberry Pi ot Arduino

Live Temperature and Humidity Monitoring over Internet using Arduino and ThingSpeak 2016

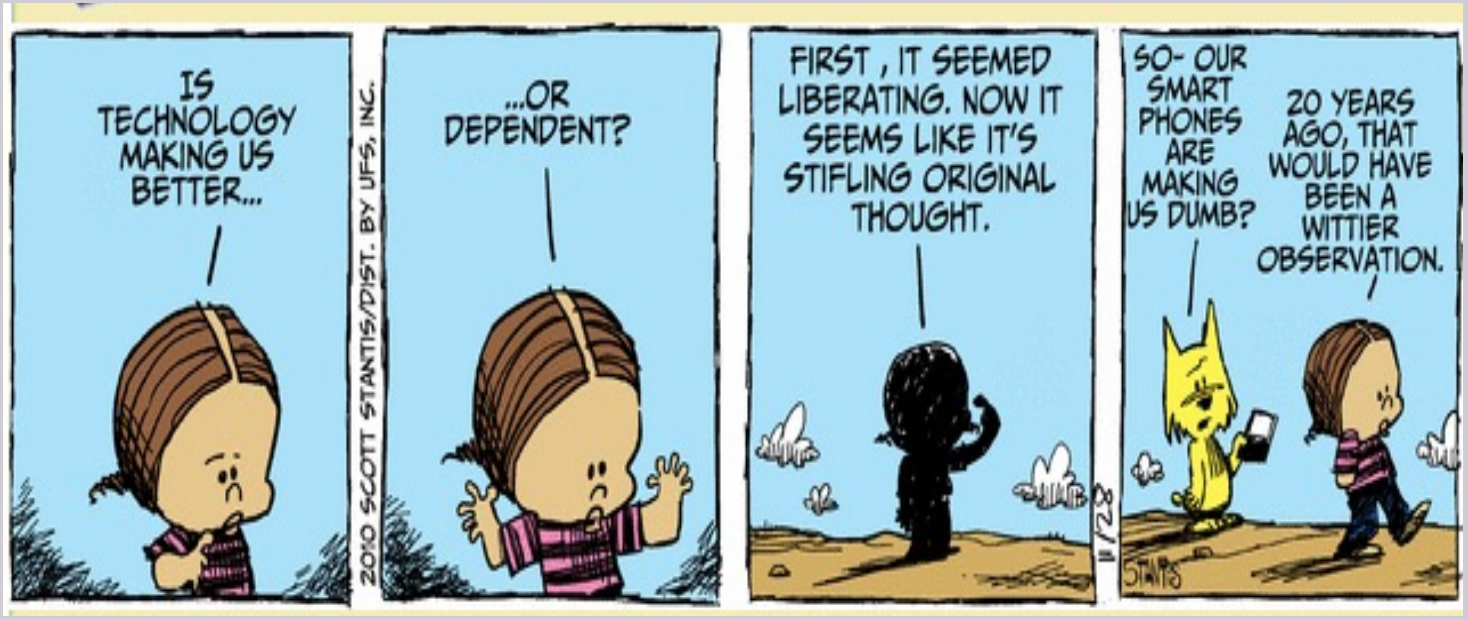
https://circuitdigest.com/microcontroller-projects/iot-temperature-humidity-monitoring-using-arduino

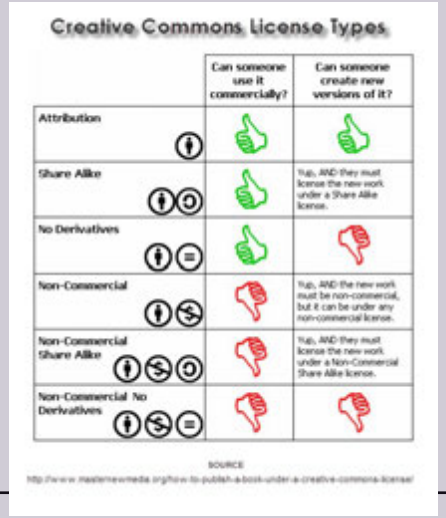
Build Your First IOT With a Raspberry Pi, DHT11 Sensor, and Thingspeak

https://www.instructables.com/Build-Your-First-IOT-with-a-Raspberry-Pi-DHT11-sen/



https://github.com/pubnub/workshop-raspberrypi





This seminar was developed by:
Rob van der Willigen
LivingLab AI & Ethics
Hogeschool Rotterdam
November 2021


the functional art
TECHNOLOGY TRENDS TO WATCH
Christian Leborg Visual Grammar
the analytics revolution
Bill Franks
generative art
MADE to STICK
EMPATHIC THINGS
THE NEW INDUSTRIAL REVOLUTION
CHRIS ANDERSON
TV AND VIDEO
THE DISRUPTION MACHINE
Learning to Program with MATLAB
BUILDING GUI TOOLS
R Data Analysis
A giant cage
10 TRENDS FOR 2014
resonate
Design for Information
Cool Infograph
The VISU
PHIL SIMON
OUR WORK HERE IS DONE
VISIONS OF A ROBOT ECONOMY
INVENT TO LEARN
10RE SECRETS with Big Data Analytics
HACKATHO
OPEN HAVE
1 VOLLE DAG
7 SEPTEMBER
WERELDHAVENDAGEN 2013
the myths of innovation
O'REILLY
THE WALL STREET JOURNAL Guide to Information Graphics
Customer Journey Mapping
Wearable Technology Design
The Hidden Pitfall in your Design
Making Things See
TEACHING with STICKS
The Internet of Things
Making Things Talk
Port Management Program
WebGL
MAKERS