Data, big and small

Introduction

This course is about managing big data. But what is big data? Why do we care about it? And why do we need to learn how to manage it?

This week we answer these questions, and learn some of the concepts that are foundational to managing big data.

We will look at:

- The different types of data format and their degree of structure
- The units that are used to measure data volume, and what they mean
- When data counts as big data, in terms of volume, variety, and velocity
- The value of data to an organisation, especially the value of big data
- Ways of storing data (relational vs non-relational databases, single machine vs cluster of machines) and when to choose one over another
- Ways of accessing data (SQL vs other techniques)
- The challenges involved in storing and accessing big data

Data format and structure

We'll be talking a lot about data - individual items of data, such as text files, and whole collections of data, such as an organisation's databases. Before doing so it will be helpful to understand a bit about the various formats in which data can be stored.

Below are some of the most common formats. They differ in how much structure they impose of the data. You might hear them being classified into three types:

- Unstructured
- Semi-Structured
- Structured

But they really fall on a continuum, from low to high structure, as we'll see.

Photo, audio recording, video recording

Suppose you find a shirt that you like while shopping, and want to remember it for later. You might take a photo of the tag. Here you have stored some data in a photo. Similarly, you might keep data about an interview in an audio recording, or data about how to swing a golf club in a video recording. These data storage formats all have very little structure, if any.

Free text

Writing things into a text file is a very common way to store data. Again, this format has very little structure.

Email

We send a lot of information in the body of emails, and we can think of an email as a format for storing data. An email has some structure - it is separated into to, cc, bcc, subject, and message parts - but within these parts there is very little structure.

Comma-separated values (CSV)

CSV format is a more structured way of storing data. The data is split into **records**, and each record has **fields**. Records are separated by line breaks (i.e. each record starts on a new line), and fields are separated by commas. The first row is optionally used to store the names of the fields - this is called a **header row**. Here's an example, which includes a header row:

```
firstName,lastName,dob,email
Jane,Johnson,1968-06-24,jane34@readymail.com
Peter,Simons,1989-10-01,p.simons@goget.com.au
```

Line breaks and commas have special meaning in this format - they are used to identify the structure

of your data. If a value contains a comma (one or more) then the value needs to be **escaped**, and the convention is to enclose it in double quotes. So the value `Hello, world` would be stored as `"Hello, world"`. This convention means that double quotes also have special meaning in CSV format. If a value contains a double quote then it must also be escaped, and the convention is to do so by replacing it with two double quotes. So the value `aka "Big John"` would be stored as `aka ""Big John""`.

Tab-separated values (TSV)

This format is very similar to CSV format, except that values are separated by tabs rather than by commas. Here's the previous example in TSV format:

```
firstName    lastName    dob    email
Jane    Johnson    1968-06-24    jane34@readymail.com
Peter    Simons    1989-10-01    p.simons@goget.com.au
```

Tabs have special meaning in this format, so any value that contains a tab (one or more) must be escaped, and the convention again is to use double quotes. Commas do not have special meaning, so do not need to be escaped; double quotes are escaped in the same way as for CSV.

Excel spreadsheet

As you will be aware, this is a very common format for storing data - organisations are usually awash with Excel spreadsheets. This format is similar to CSV and TSV, but can contain more information about structure, formatting, and calculations (for example, it can contain information about alignment and shading, and calculated fields).

Hypertext markup language (HTML)

This is the format in which the data in a webpage is stored. Items of data are enclosed in **tags** - an **opening tag** and a **closing tag**, which demarcate the boundaries of the item, and indicate something about the nature of the item. For example, a paragraph of text is stored as `<p>text goes here</p>`. The opening tag is `<p>`, the closing tag is `</p>` - and these indicate that the item is a paragraph of text, and they mark the start and end of the text. Tags can contain other tags, so HTML files can have a hierarchical structure. An HTML file can contain quite a bit of structure. Here is an example:

```
<html>
  <head>
    <title>Home page</title>
    <link rel='stylesheet' type='text/css' href='mystyles.css' />
    <script src='myscripts.js'></script>
    <style>
      p {margin: 0}
    </style>
  </head>
  <body>
    <h1>Hello</h1>
```

```
    <p>Hello world!</p>
  </body>
</html>
```

Extensible markup language (XML)

You can think of XML as being a more general version of HTML. Data is organised in a hierarchy of opening tags and closing tags. Here's the same data that was stored as CSV and TSV above, stored in XML format:

```
<xml>
  <person>
    <firstName>Jane</firstName>
    <lastName>Johnson</lastName>
    <dob>1968-06-24</dob>
    <email>jane34@readymail.com</email>
  </person>
  <person>
    <firstName>Peter</firstName>
    <lastName>Simons</lastName>
    <dob>1989-10-01</dob>
    <email>p.simons@goget.com.au</email>
  </person>
</xml>
```

Javascript object notation (JSON)

This is similar to XML, although is less "verbose" - it contains less language, by relying on conventional uses of square brackets `[ ]` and curly brackets `{ }`. Here is the data from above, stored in JSON format:

```
[
  {
    "firstName": "Jane",
    "lastName": "Johnson",
    "dob": "1968-06-24",
    "email": "jane34@readymail.com"
  }
  {
    "firstName": "Peter",
    "lastName": "Simons",
    "dob": "1989-10-01",
    "email": "p.simons@goget.com.au"
  }
]
```

Relational database

In a relational database, data is stored in a highly structured set of tables, each of which rows and columns, which can be connected to each other in the interests of efficiency and data integrity. You will learn quite a bit about this format in Weeks 1-3. Here is the data from above, stored in a relational database:

Data volume

We'll be talking a lot about data volume as well. So it's worth clarifying the units that are used to measure volumes of data. The numbers involved can get very large. To help grasp what they mean, we've converted each unit into a more meaningful approximate equivalent.

bit (b)

A bit is either 0 or 1. When data is stored digitally it is ultimately stored as a sequence of 0s and 1s.

byte (B)

A byte is 8 bits. Since each bit can have one of two values (0 or 1), a byte can hold 2^8 = 256 different values, which is enough different combinations for us to use one byte to store **one character**.

kilobyte (kB)

A kilobyte is 1000 bytes (more accurately, it is 2^{10} = 1024 bytes, but it is common to use 1000 instead). A kilobyte is about **a paragraph** of text (a typical paragraph has about 1000 characters).

megabyte (MB)

A megabyte is 1000 kilobytes (more accurately, 1024 kilobytes). A megabyte is about **a large book** of text (a typical large books has about 1000 paragraphs - about 3 per page, for about 333 pages).

gigabyte (GB)

A gigabyte is 1000 megabytes (more accurately, 1024). That's about **a small bookshop** of text (a typical small bookshop has about 1000 books).

terabyte (TB)

A terabyte is 1000 gigabytes (more accurately, 1024). That's about **a university library** of text (a typical university library has about 1000 small bookshops of books).

petabyte (PB)

A petabyte is 1000 terabytes (more accurately, 1024). That's roughly the amount of text in **all the books in Australia** (about 1000 libraries worth).

Big data

In this course, we are especially interested in big data. So what is big data?

Any set of data can be measured along three dimensions:

- **Volume**: how much disk space it requires, measured in gigabytes, terabytes, petabytes, etc.
- **Variety**: how diverse it is in kind - spreadsheets, text files, images, graphs, etc.
- **Velocity**: how quickly it accumulates or changes.

At one extreme there is data that is small on all three dimensions. E.g. an Excel spreadsheet containing a cafe's employee details - it doesn't require much disk space, it is uniform in kind, and it changes very slowly. This is a clear case of small data.

At the other extreme there is data that is big on all three dimensions. E.g. Facebook post data - it requires a lot of disk space, it includes data of various kinds (via attachments), and it changes very quickly. This is a clear case of **big data**.

In between there are many cases that are not clearly big and not clearly not big - these are cases in which it is not clear whether the data is big data. There are lots of cases like this. Nevertheless, because there are also plenty of clear cases, the concept of 'big data' is still a useful one, and throughout this course we'll be considering the clear cases.

Using big data

Organisations are increasingly discovering that big data is **valuable**. By using big data they can do such things as:

- Better understand and predict the behaviour of people
- Better identify new opportunities for products or services
- Discover more efficient and effective ways to operate
- Reduce costs and wastage

(You will see examples in this week's assessment task). Because of this, organisations are increasingly gathering more data - not only their own internal data, but also external data such as census data.

The data by itself is just data - it's not helpful until we make it **visible**, which means presenting it in meaningful and digestible ways (as graphs, for example). We need to find the story behind the data.

Finding this story can be difficult. It is sometimes said that big data is **viscous** - it takes time to clean and analyse the data.

And we need to be careful when gathering and using big data:

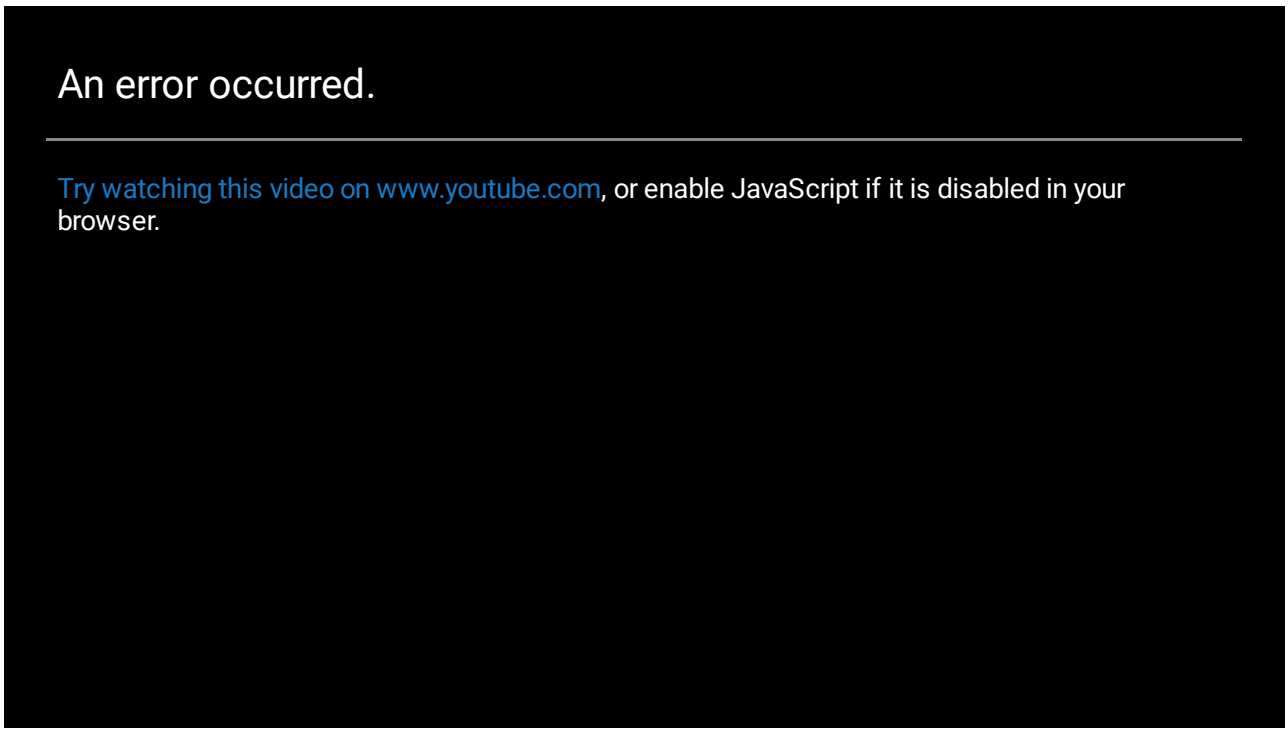
First, we must care about **veracity** - how accurate it is. If your data is not accurate then it will not be so helpful, and in fact might even be harmful, if you treat it as if it were accurate.

Second, we must keep in mind that the usefulness of big data is **volatile**. Data that was, at one time, relevant and useful, might no longer be relevant or useful at another time, and should be excluded from analysis. For example, if a large organisation sells one part of its operations then the data pertaining that part might just muddy any further analysis of the organisation's data, and might best be excluded.

Third, we must be aware that the meaning of our data is **variable** - it might change over time, or with context. An example is the processing of natural language, such as tweets or forum posts. Language varies from location to location, and changes over time, so when processing text to extract meaning we need to be aware of the context, and keep re-evaluating our techniques to make sure they still apply.

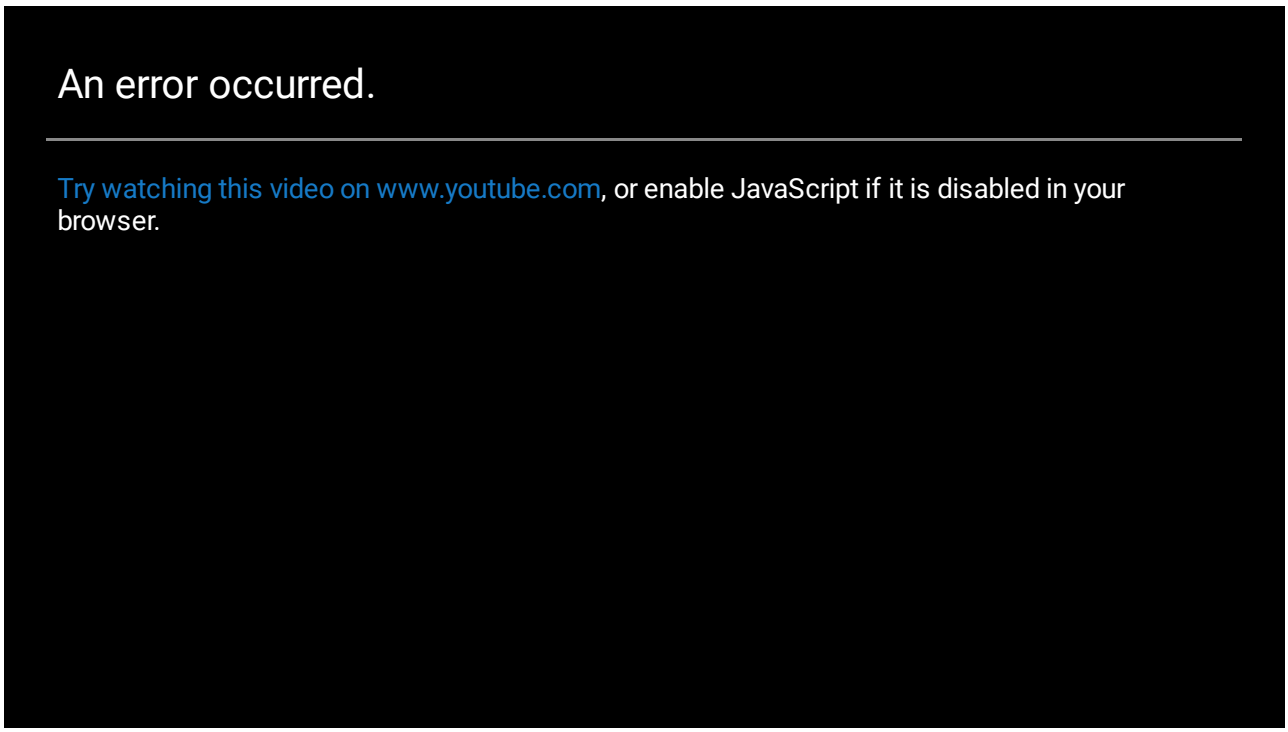
In the previous slide we saw that the size of data can be measure along three dimensions - volume, variety, and velocity. These are sometimes called **the three Vs of big data**. The words in bold face above mark other concepts that are important to the management of big data, and are sometimes thought of as **further Vs of big data**.

Google data centre (1:46)



This video shows how much infrastructure is involved in storing and processing big data - it's a tour through a Google data centre.

Big data and its uses (2:46)



This video gives a concise overview of some of the issues involved with using big data.

Databases

No matter what kind and volume of data you have, you will want to store it in a database, or perhaps several databases.

A database is just a store of data. You already have your own database, and manage it. When you store files on your computer, organise them into folders, and delete ones that you no longer want or need, you are managing a database.

There are different kinds of databases, and which one(s) an organisation should use depends on the nature of their data (its kind and size) and what they want to do with it.

One main division is between **relational databases** and **non-relational databases**.

In a relational database your data is kept in a set of pre-defined and highly structured **tables**. This is a good way to keep tight control over your data. Within relational databases a distinction is drawn between **operational databases**, which are good for fast and frequent data updates and querying, and **data warehouses**, which are good for storing and analysing large amounts of data that doesn't need to be accessed quickly, or updated often. Relational databases need to be designed carefully, and how that's best done differs slightly between operational databases and data warehouses. When it comes to working with relational databases, it's all done primarily using the language SQL (with the help of GUIs to construct SQL). You will learn about all of this in Weeks 2 and 3.

A non-relational database is any other kind of database - one in which the data is not kept in tables. It is sometimes called a **data lake**. Rather than having tables with rows of data, in a data lake you typically have collections of documents, and the documents might be of various kinds (text, CSV, pdf, etc.). So data lakes are less structured than relational databases. The system of folders in which you keep files on your computer is a good example - you don't have tables in which you store your data, you just keep a disparate collection of files of different formats

Why use a data lake? Why not keep all of your data in a relational database?

First, it takes a lot of work to set up a relational database, and a lot of work to put your data into it. Imagine trying to extract all of the data in your personal files and putting it into a system of tables - it would be an enormous amount of work. That work might be worth it, if you want to do a lot of careful analysis of it. If not, then it might not be worth the effort.

Second, it might not be practically possible. Some data cannot be put into tables - the data in a photo, for example. Or the data might be collected too rapidly. Or there might be too much data. A relational database is limited in how much data it can hold. If you exceed that limit, then you'll have to use a non-relational database instead.

Data lakes can contain an enormous amount of data - too much to be stored on a single machine. In this case the data must be stored across multiple machines in a cluster (some organisations have

clusters containing thousands of computers).

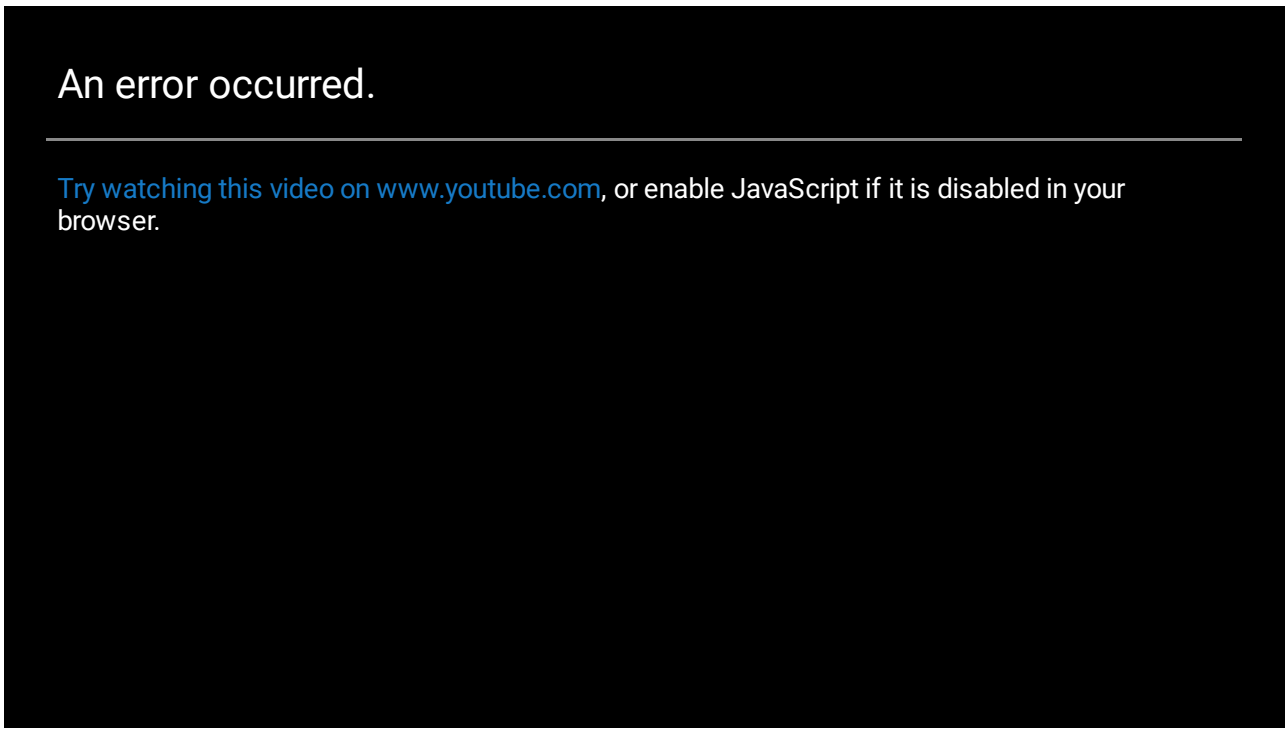
Working with data spread across multiple computers requires special techniques - special techniques for storing it, and special techniques for retrieving and analysing it.

Many organisations use a storage system called **Hadoop** to work with their distributed data, and a process called **MapReduce** to access and analyse this data. You will learn about Hadoop and MapReduce in Week 4.

Along with Hadoop, a lot of different pieces of software have managed to help make many aspects of working with Hadoop more user friendly, or effective. You will learn about two of these, **MRJob** and **Hive**, in Week 5, and a third, **Spark**, in Week 6.

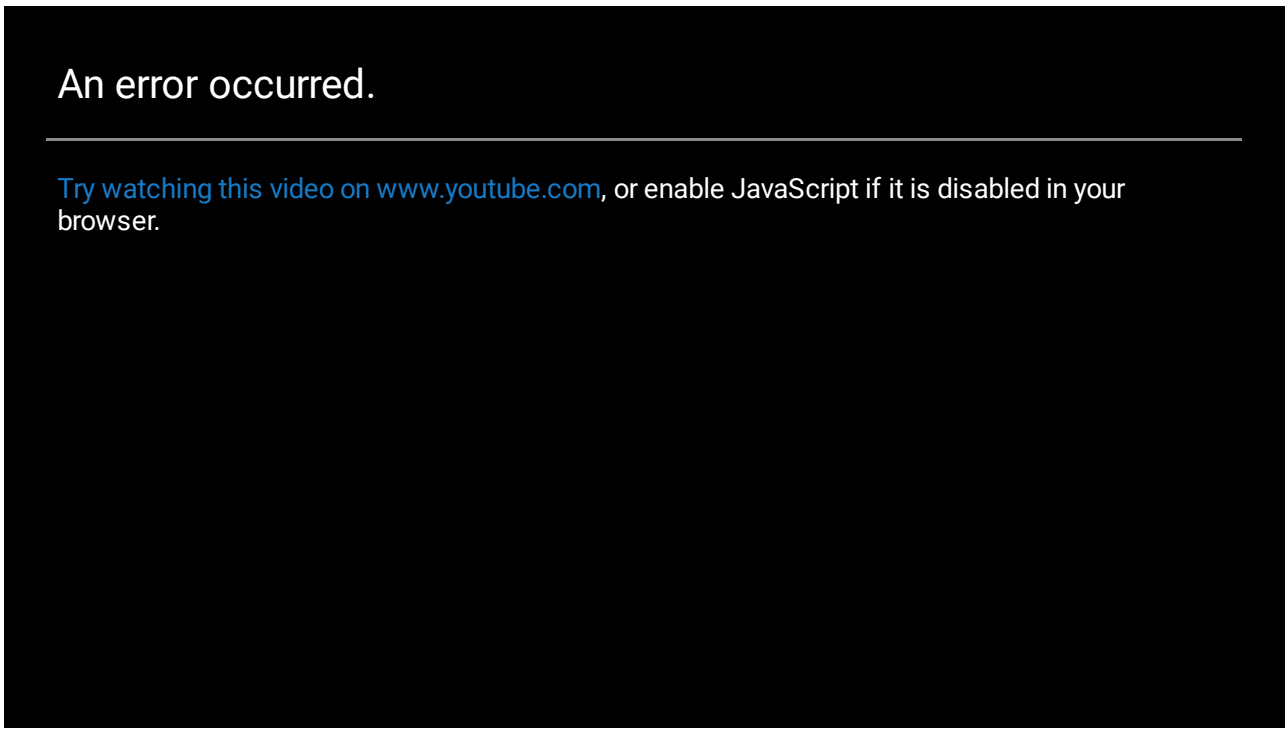
A number of these have found ways to use SQL to work with big data, because SQL is easy and popular. The data being accessed is not actually stored in a relational database - it's too big. But it can be used as if it is. Thus the importance of learning SQL - it's the main tool for working with small data, but also one of the main tools for working with big data.

Relational vs non-relational databases (21:30)



Should your data be stored in a relational database, or not? This video considers some of the pros and cons of each. When you see or hear "SQL", think "relational database". When you see or hear "NoSQL", think "non-relational database", or "data lake".

Choosing a database (21:53)



Like the previous video, this one considers some of the pros and cons involved in choosing which kind of database to use for your data.

Facebook and eBay and big data

Here are a couple of articles about how Facebook and eBay are using big data. They contain quite a bit of technical terminology, but you don't need to understand that - just focus on the general ideas that the articles are conveying.

Article about how Facebook is deploying big data

Article about how eBay uses big data

Further resources

If you're interested in the broader issues to do with our use of big data then you might enjoy the following books (this is just further reading, not required for this course):

Big Data, by Viktor Mayer-Schonberger and Kenneth Cukier (2013). Gives an overview of the benefits and dangers of using big data.

Platform Revolution, by Geoffrey G. Parker, Marshall W. Van Alstyne, and Sangeet Paul Choudary (2016). Considers some of the impacts that platforms based on big data are having on our organisations and our lives.

Weapons of Math Destruction, by Cathy O'Neil (2016). Highlights some of the dangers of not being careful about how we use algorithms to analyse big data.

Everybody Lies, by Seth Stephens-Davidowitz (2017). Gives examples of what we can learn by analysing big data, especially internet searches.