Formal data management lifecycles

CA682

In the first week, I introduce a **generic data analytics** pipeline for CA682 (shown below) outlining the main tasks that occur in a data-driven project.

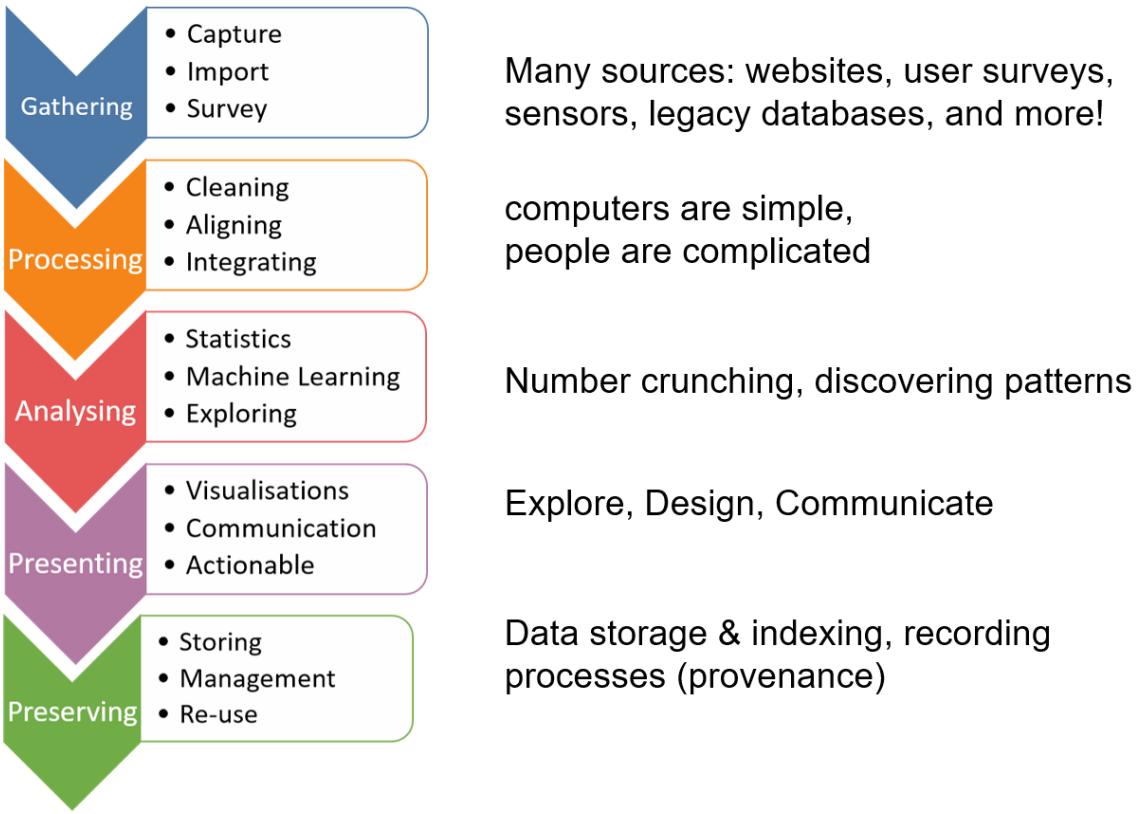


Figure 1: A Generic Data Analytics Pipeline

However there are many formal data management systems used in different domains. Let's look at a few examples:

- A research data management lifecycle
- Domain example: US Geological Survey Data lifecycle
- Enterprise Data Management
- Data mining: CRISP-DM

At the end of this document are some questions for you to reflect on and discuss.

A research data management lifecycle: UK Data Service

A key component of **data-driven research** (whether in sociology, economics, medicine or artificial intelligence) is the data process. The UK data service has lots of materials to support research projects in effectively managing their data available here.

Watch the video at https://youtu.be/-wjFMMQD3UA (it has no sound) to explore this data management lifecycle.

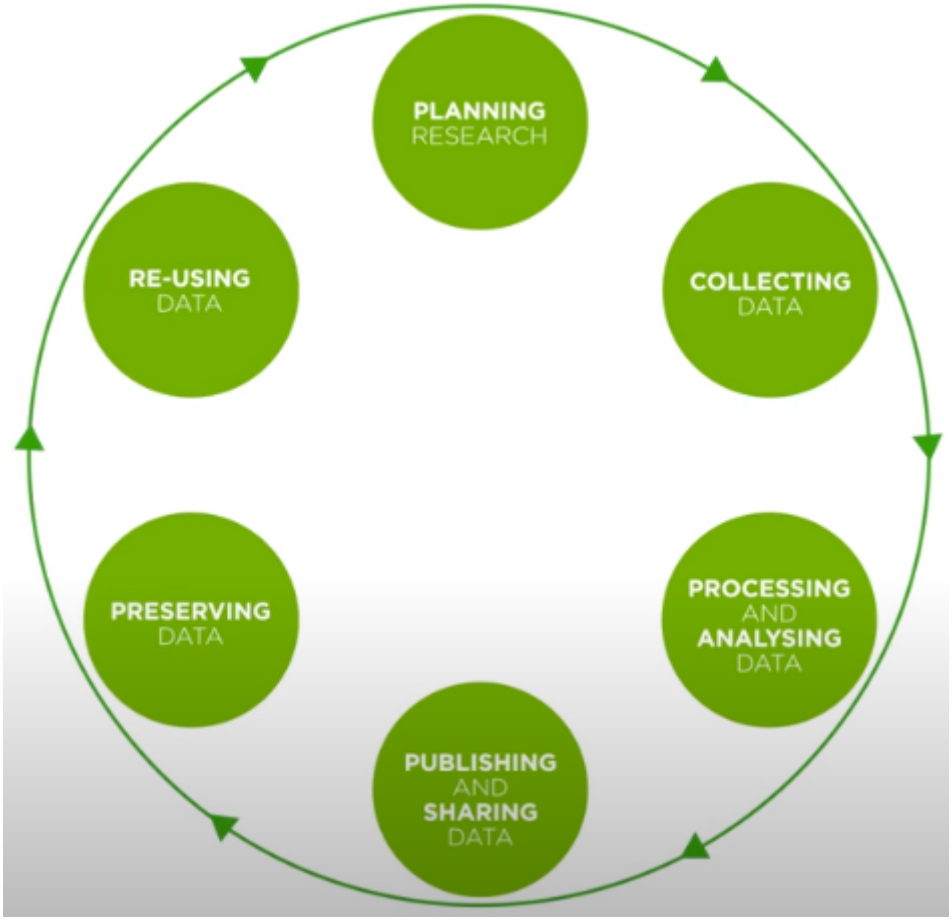


Figure 2: UK research data management lifecycle

A key component of all research is the importance of **citing your sources**. Watch this Youtube video from the UK data service on citing data - Cite Your Data: https://youtu.be/cEGYw19FYn8

US Geological Survey Data Lifecycle

Domains can have specific data requirements. Check out the data lifecycle from the US Geological Survey.

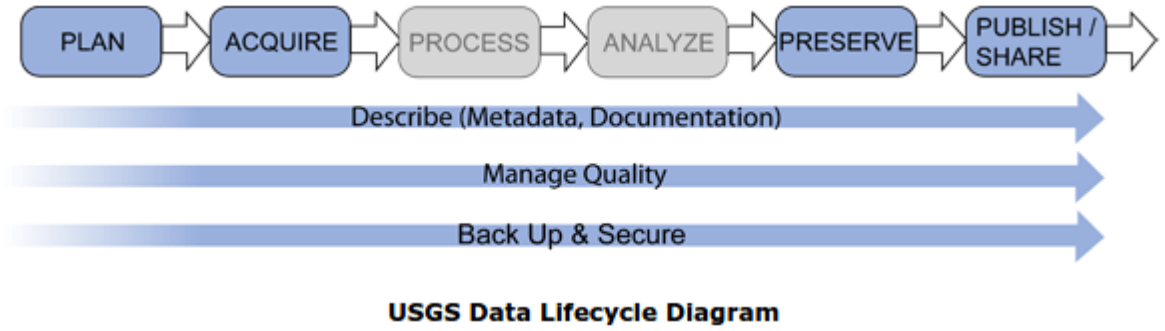


Figure 2: USGS Data Lifecycle Diagram

What differences can you see from the general UK research data lifecycle? Notice just how much material and support resources are required to document the lifecycle!

Enterprise Data Management

Of course, data is also a key component of many business processes. For example, Figure 4 below provides a model for Data Maturity (software engineers compare this to the Capability Maturity Model) from CMMI Institute. Note that many enterprise solutions are commercially funded and supported so access to details is often limited unless you are a client of the provider.

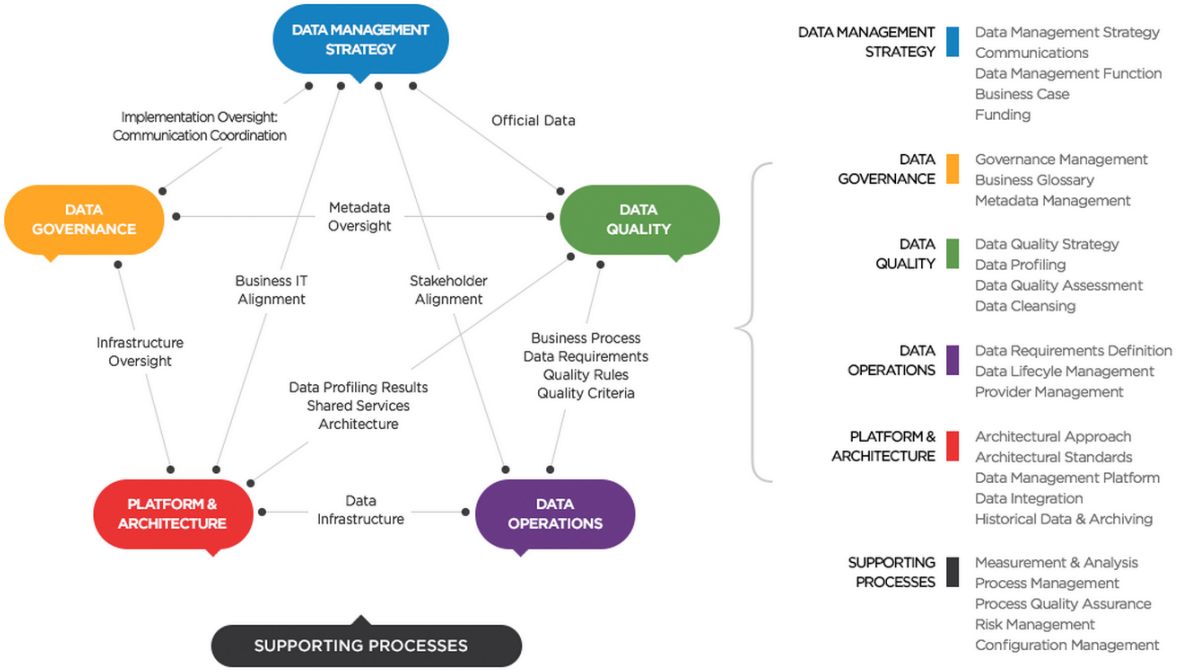


Figure 4: Model for Data Maturity, (From: CMMIinstitute, https://cmmiinstitute.com/getattachment/cb35800b-720f-4afe-93bf-86ccefb1fb17/attachment.aspx

Another example of an enterprise model comes from the MITRE Corporation as seen in Figure 5 below. Note the variety of tasks and the range of skills that would be required to support this!

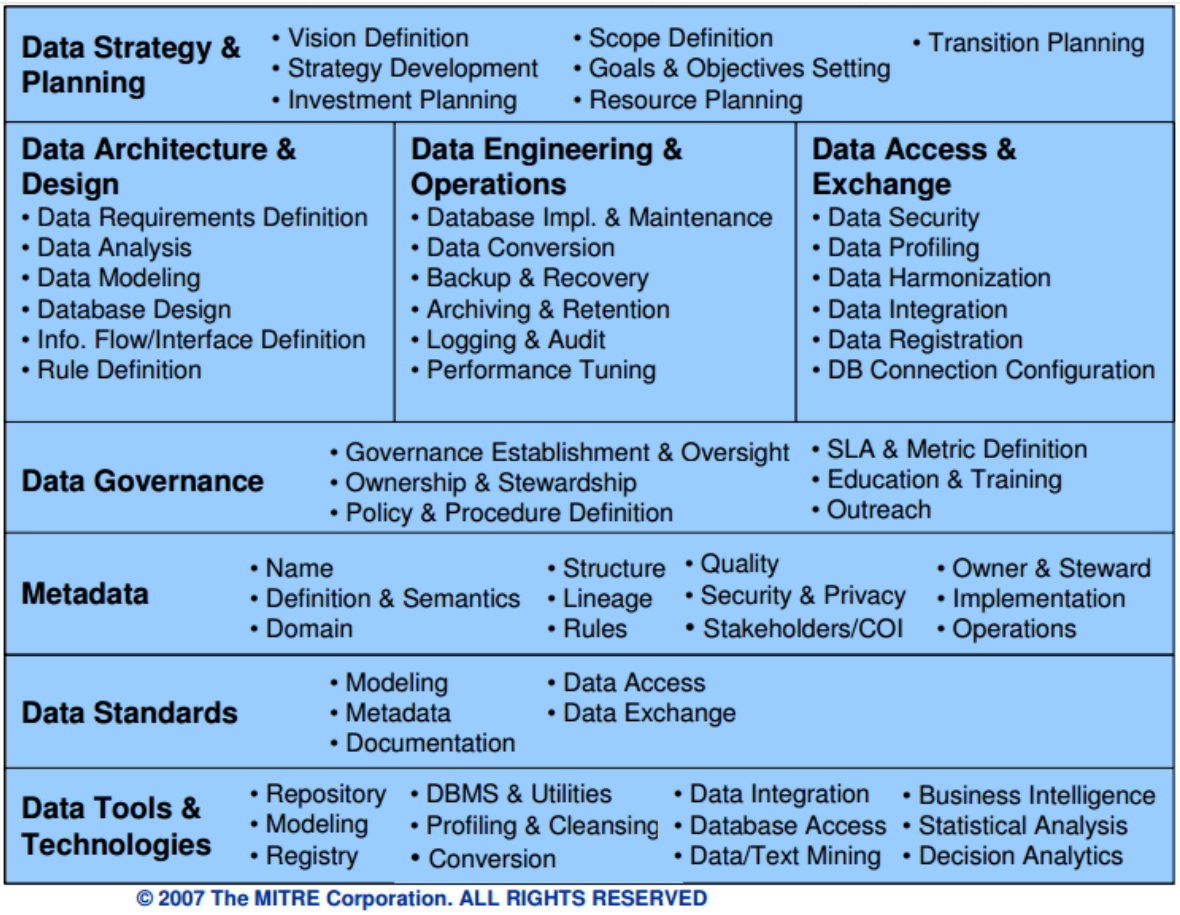


Figure 5: Enterprise Data Model (MITRE Corporation), Source: https://dama-ncr.org/Library/2007-03-13Brooks.pdf

CRISP-DM (data mining)

Finally, you may have already heard of CRISP-DM, an open standard for data mining processes that is widely used. This is also a data analytics lifecycle. Notice how, unlike the other cycles we've seen, the sequence of phases is not as strictly defined in CRISP-DM.

Figure 6: Different phases of CRISP-DM

Exercise

Discuss: Compare one of these models to the generic pipeline. What extra tasks or phases are present? What tasks or phases are missing?

Reflect: In your own work or research what does the data management lifecycle look like? What parts are you concentrating on? Are you missing any important steps?

Notes on Files

Suzanne Little, suzanne.little@dcu.ie

Files of one form or another are the most common way that computers store data. In fact from a computer's perspective, most content is a file.

The different file formats, sometimes indicated by the extension, are processed in different ways depending on the programme and their purpose. When you have data in files, it's good to consider whether the file is:

1. Text or binary.
2. Open or proprietary.
3. Structured or unstructured.

A simple way to distinguish between **text** and **binary** files is to open the file in a text viewer like Notepad or Gedit. In Figure 1 below, you can see a screenshot from trying to open a binary file (a PDF file) in a text viewer, it looks like complete nonsense! But the proper programme (a PDF reader like Acrobat) is able to interpret the binary data and render or show the PDF document. Text files are easier to read using programmes and libraries. A binary file is simply a non-text file and common file formats include PDFs and older Microsoft Office documents (note: newer Microsoft Office documents use an XML based text format).

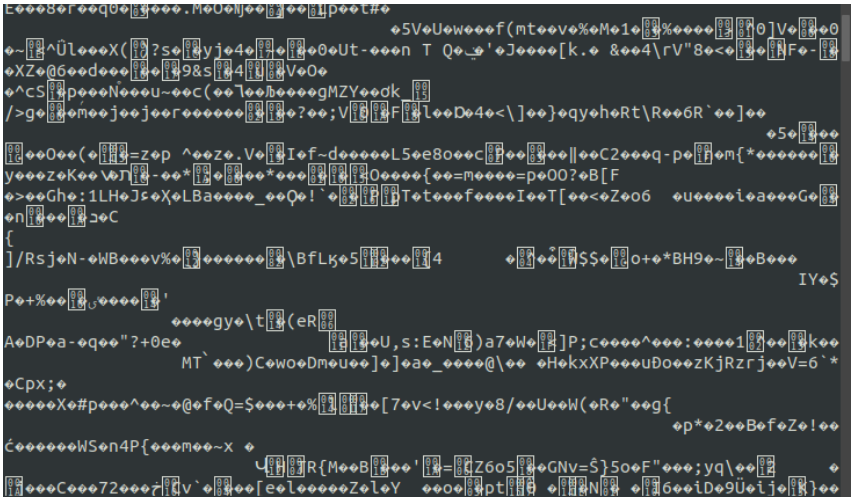


Figure 1: Screenshot of binary file open in text viewer

Another consideration when looking at data files is to identify those that may have a **proprietary** format. Proprietary formats belong to a company and usually require a specific programme to read and manipulate the contents.

The most common **structured** or tabulated data format is CSV (comma separated values). This is a really useful format for data interchange as it is very simple and widely available as a format for reading or writing data from most programmes. There are limits with CSV and we will take a look at some of these in the exercises at the end of this section. You might

also come across TSV or other variations where there is a different separator character used. Can you find out what TSV stands for?

Other text-based structured data formats that you should be familiar with include:

- JSON: JavaScript Object Notation is often used in web-based APIs and is a compact way to record an object and attribute-value pairs.
- XML: (eXensible Markup Language) is an abstract format to build structured text documents based around tags (e.g., \<names\>) to mark up a document. There are many common XML-based formats (e.g. YAML, ATOM, RSS) and most programs can read or export these formats.
- [KML](https://developers.google.com/kml/documentation/kml_tut?csw=1): (keyhole markup language) a specific XML-based file format for geographic data used by Google in Google Maps.
- (X)HTML: the format used for web pages is more concerned with how to structure and view data (text, links and images) but sometimes you need to process data that may only be available in HTML format.

Many database programs (such as SQLite, MariaDB and PostgreSQL) may use text files to store the data, e.g., DB, SQL. The programs are optimised to process, store and retrieve from these files but often they are text-based and can be viewed and read by other programs.

There are other specialist data formats (for example, GDP and ASX), some of which may require a specific programme or library to read.

You might like to take a look at the [Wikipedia list of file formats](https://en.wikipedia.org/wiki/List_of_file_formats) to see just how many different variations there are!

Reference Sheet for Data Types

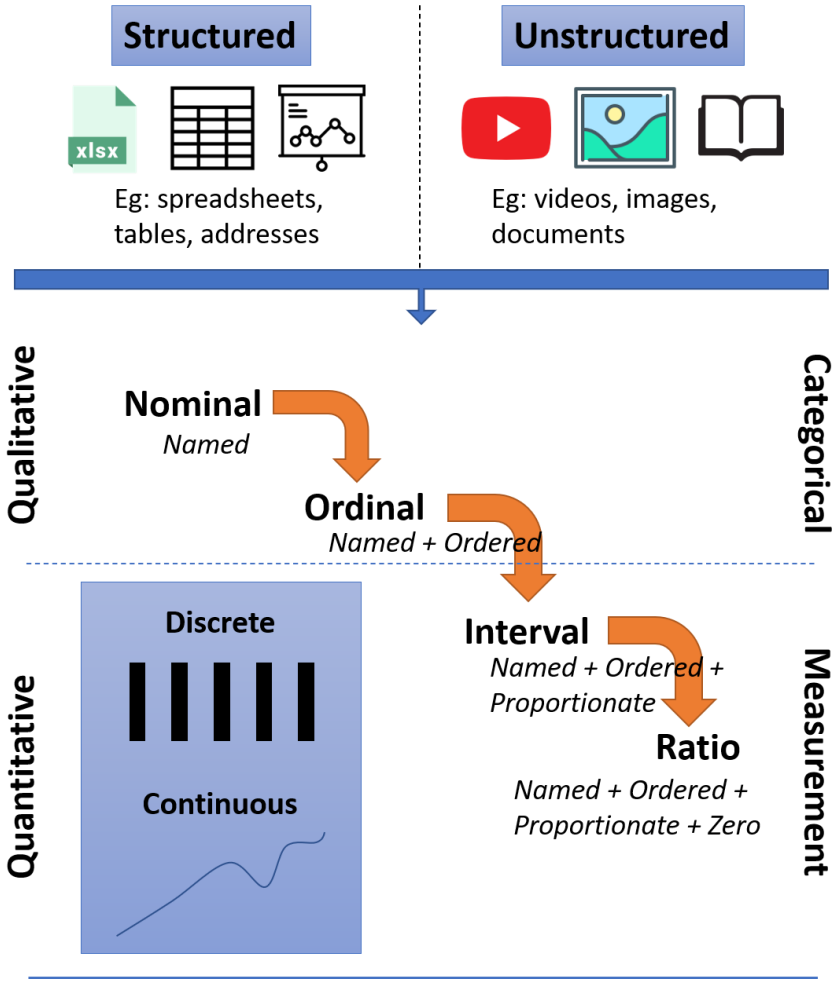
Suzanne Little, suzanne.little@dcu.ie

You may have seen data types before if you've built a relational database and specified the SQL data types or chosen the column type in a spreadsheet. These data types like string, integer, date etc. are very specific and normally determine the storage requirements, available operations and presentation format for the underlying data.

Here we will look at four different ways of categorising data features plus a few special cases:

- Structure vs Unstructured
- Quantitative vs Qualitative
- Discrete vs Continuous
- NOIR levels of data
- Special types - time, space, documents & media

Figure 1 below provides an overview of the relationship between these terms for talking about data.

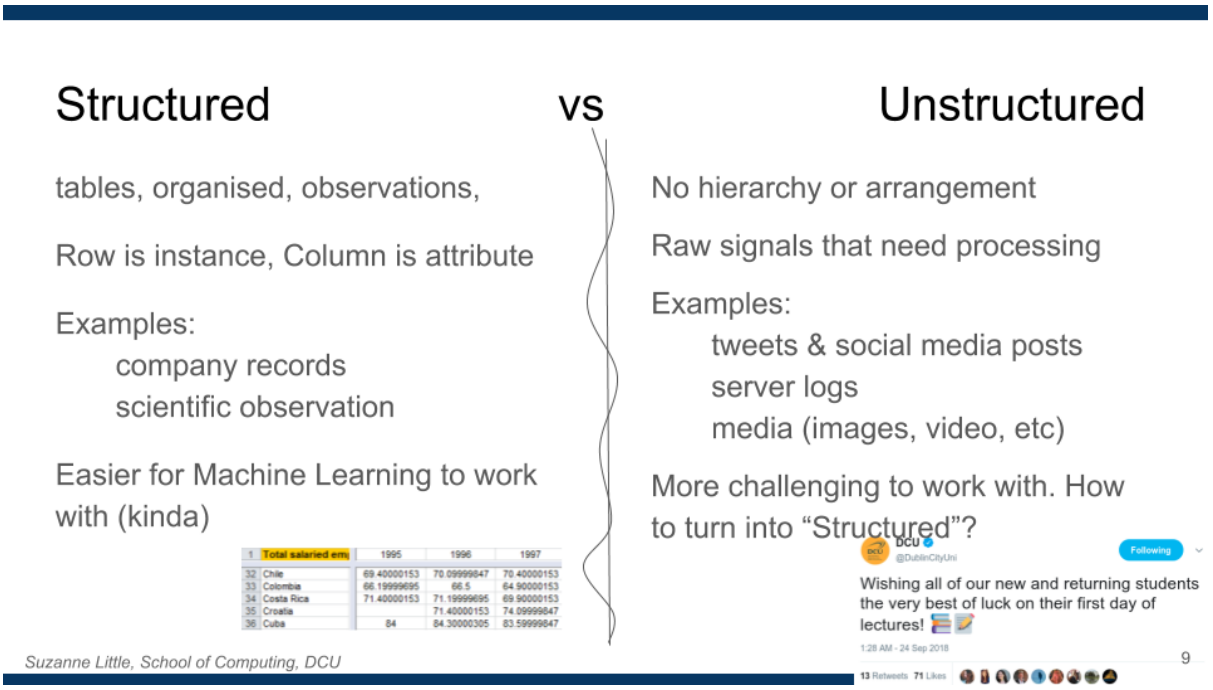


Why is this useful? The type of data in a dataset determines:

- What statistics are possible/meaningful
- How data can be processed and/or stored
- Which machine learning model can be used
- Which visualisation method to use

References

Structured vs Unstructured



When we look at where data comes from there are two categories that we can assign - structured and unstructured.

Structured data, as the name suggests, has been formatted into an orderly arrangement. This might be a table or tabular format like shown in Figure 1 below. Here we have rows or instances, one for each country and columns or attributes, one for each year. Simple tables like this are common from domains like company records or scientific observations. You can also find more complicated tables that are multi-indexed. In the example here you might want a multi-indexed table if you have more than one value to record for each year, for example, total salaried employees and median wage.

1	Total salaried em	1995	1996	1997
32	Chile	69.40000153	70.09999847	70.40000153
33	Colombia	66.19999695	66.5	64.90000153
34	Costa Rica	71.40000153	71.19999695	69.90000153
35	Croatia		71.40000153	74.09999847
36	Cuba	84	84.30000305	83.59999847

Figure 1: Example of structured data in a spreadsheet

The advantage of structured data is that it is generally easier for machine learning applications to work with. There are common storage formats that many tools are able to load, read and save and the data is normally cleaner (Although we'll learn later that this is not always the case!).

In contrast to structured data, we have **unstructured data**. Unstructured data has no hierarchy or arrangement and usually comes from raw signals that need to be processed in some way. Common examples of unstructured data include tweets (See Figure 2) and social media posts and multimedia data such as images, video and audio. Data sources like server logs and user interactions can also be unstructured though they do have some order.

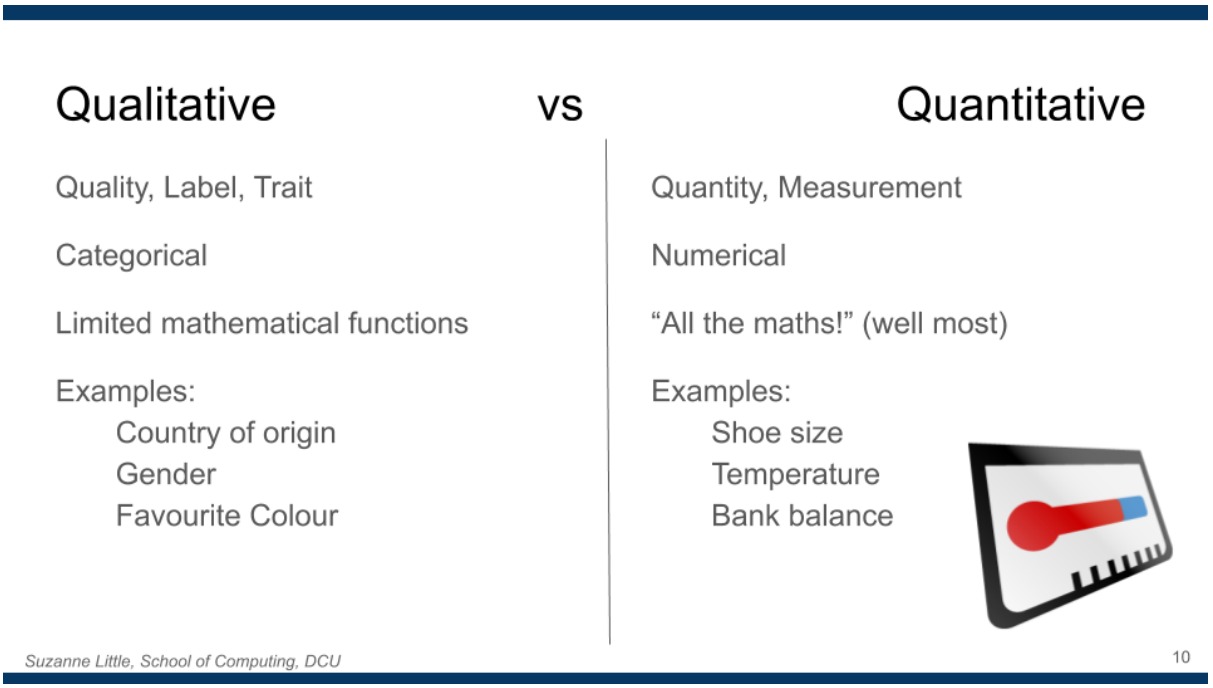
Figure 2: Example of tweet

Unstructured data is rich and challenging. The rise of the Internet, consumer digitisation devices, social media, mobile devices and wearable sensors (like Fitbits) mean that we are producing more data than ever before. Using unstructured data for analytics, machine learning or AI requires that the data is processed and converted into a usable format or structure. We'll see later that unstructured data forms the basis for what is termed "big data" and the growth in opportunities to build data-driven services.

Deciding on whether a dataset or data source is structured or unstructured can depend somewhat on how you plan to use that data, do you want to analyse the *content* of the tweets or just to *count* how many people liked it? When you approach a project, determining if you are working with primarily structured or unstructured data can help to estimate the amount of data processing work that could be required.

Quantitative vs Qualitative

These terms are often used because of their alliterative properties (which make it hard for me to pronounce them!) but the distinction here is between labels (strings) that describe a quality and numbers that describe a quantity.



Qualitative data is normally stored as text or a string and describes a label or trait of the instance. This is also called categorical data types. Some examples of qualitative features from a data set about this class might be the country of origin, gender, eye colour of each student. A key constraint of categorical data is that there are only limited mathematical functions you can perform, normally counting the occurrences and most popular. For example, you can't calculate the average country of origin for this class!

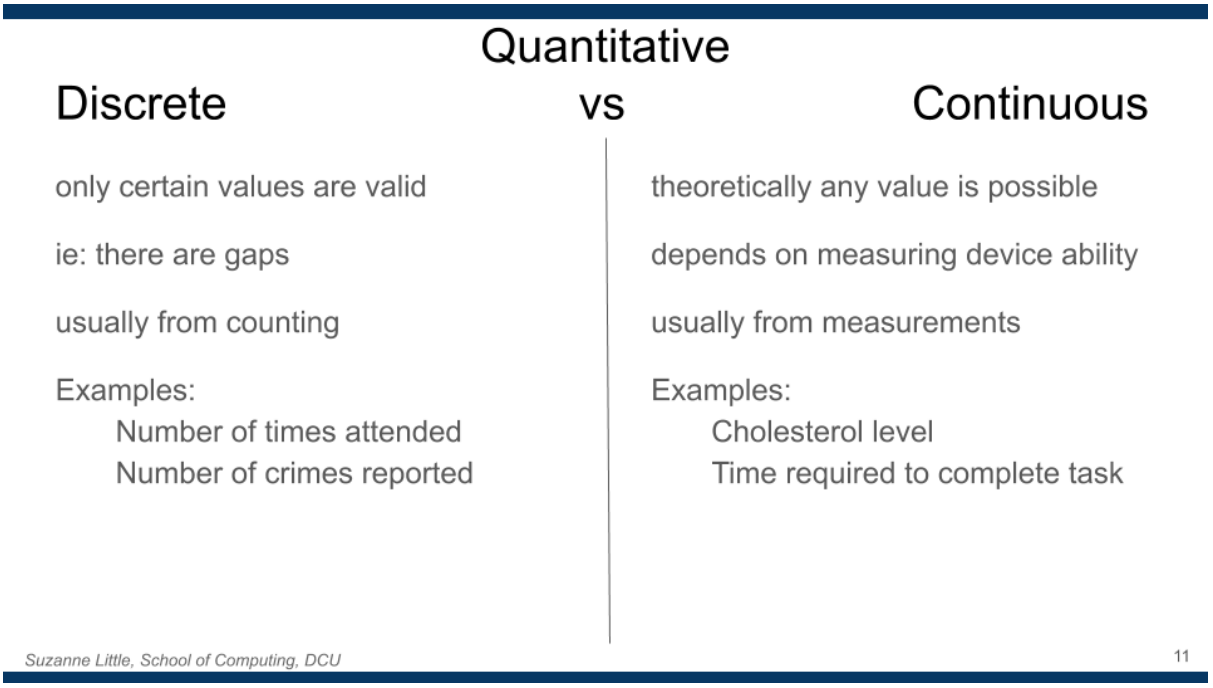
Quantitative data is numerical. These are features or attributes that give a measurement, generally stored as either an integer or real-valued (float) number. Clearly this then allows mathematical operations to be performed on the values, though as you'll see in the section on NOIR descriptions of data there are still limits on which operations should be used for numbers. Some examples of quantitative features from a data set about this class might be height, shoe size or current bank balance.

Understanding whether you have labels or measurements is useful when choosing the appropriate summary statistics for a dataset and to identify appropriate visualisation types.

Discuss: What about mobile phone numbers -- is this a quantitative value? Think about the type of maths operations you would perform on phone numbers. Is it a qualitative or quantitative feature?

Discrete vs Continuous

Let's talk about two qualities specific to numeric (quantitative) data - discrete and continuous. You've probably seen these before in introductions to statistics.



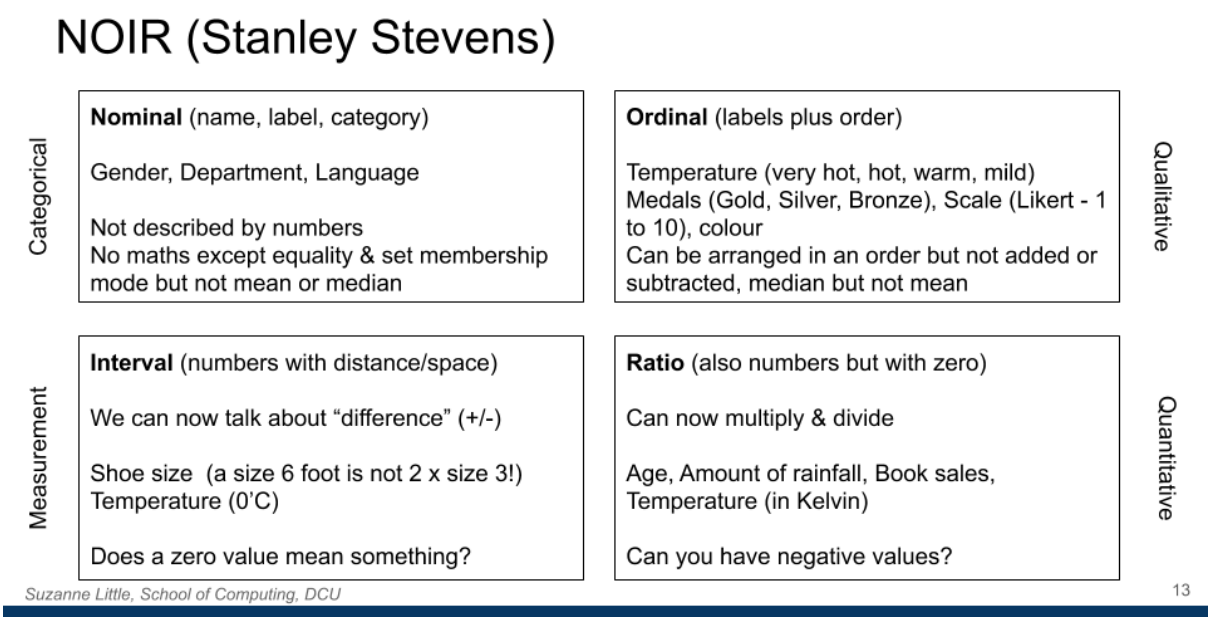
Features that have **discrete** numeric data can only take certain values. This is commonly found when you count objects, for example, the number of students in a class, the number of times they attended. These objects can't be divided so values like "100.5 students attended class" are not possible.

The opposite of discrete data is **continuous** data. With features that have continuous numeric data theoretically any value is possible depending on the ability of your measuring device. Examples of continuous data might be measuring cholesterol level or the time required to complete a task.

So a simple difference is that discrete data is counted while continuous data is measured. Again this is useful in selecting the appropriate statistical methods, machine learning models and suitable visualisation methods.

NOIR levels of data

The final way to categorise the type of data measurements comes from statistics and has four levels of classes for both qualitative (categorical) data and quantitative (numerical) data.



The Nominal, Ordinal, Interval and Ratio classes of data measurement are particularly useful for identifying appropriate statistical methods and therefore the machine learning models that may be used for a dataset. Some of the labels are a little confusing but forming the habit of thinking about how you would class each attribute in your dataset will help you identify good analytics and visualisation methods and avoid the trap of performing an analysis or visualisation that is false or misleading.

Nominal
name, label, category - not described by numbers (though may be stored as numbers)
Operations: Only equality and set membership. Can calculate mode but not mean or median.
Examples: Gender, Department, Language

Ordinal
Labels with a natural order
Operations: Can be arranged by order (sorted) but not added or subtracted, median can be calculated but not mean.
Examples: medals (gold, silver, bronze); scales (Likert 1 to 10) and temperature descriptions (cold, warm, hot)

Interval
Numbers with proportionate intervals. Negative values are possible and meaningful.
Operations: Can now use "difference between", addition and subtraction operations and can calculate descriptive statistics.
Examples: income, temperature (°C, °F) and shoe size.

Ratio
Numbers with proportionate intervals but the value of zero has meaning (eg, “absolute zero”) and therefore negative values are not possible.
Operations: Can now multiple and divide. All statistical operations are possible.
Examples: age, amount of rainfall, book sales, temperature (in Kelvin!)

Applying NOIR classifications to measurements

You don’t want to try and calculate the average of a list of people’s names or their average eye colour! Other odd comparisons might be deciding that someone with a size 10 shoe has a foot that is twice as big as someone with a size 5 shoe. Shoe size is an *interval* measurement because multiplication often doesn’t make sense.

Temperature is an interesting case. When you use terms to describe the weather (cold, cool, warm, hot) then these are labels with a natural order so this measurement is *ordinal*. If you measure temperature using a standard system like Celcius or Farenheit then it is an *interval* measurement because the difference between 10 & 20 and 20 & 30 has the same meaning (the interval is equivalent). It’s also possible to have negative degrees Celsius and Fahrenheit so a value of zero does not mean that there is *no* temperature. However, if you are using the Kelvin scale (for example, in physics and chemistry) then zero is a possible and meaningful value that means there is no physical heat! This becomes a *ratio* measurement.

A few special cases that can be debated include Likert scales and any measure that is a count of something. Likert scales refer to a type of survey question that uses a set of responses that are ordered so that one response is greater than another. For example, how are you enjoying this course? (1 - not at all to 5 - it’s amazing!). A well designed and understood Likert scale set of responses has the same difference between each answer and so you could classify it as a numerical measurement and therefore as *interval* data. However, they are often less precise and therefore work better as *ordinal* (ordered but not regular) data.

When you are counting something - number of items, number of days, etc. - this is generally a *ratio* value because you can’t get a negative answer (For example, I don’t own -3 coffee mugs) and because you can perform different descriptive statistics on the answers. However, counts can have their own statistical implications and you should refer to your courses on statistics and machine learning where necessary.

Special types of data

Now that we've looked at the main ways we can classify or discuss data such as structured or unstructured, qualitative or quantitative, discrete/continuous or NOIR, there are a few special cases that you need to watch out for.

These data types often require specialised processing or provide the opportunity to use particular visualisation methods. We will discuss these below

Temporal (or Time Series): when reviewing or gathering a dataset that has timestamps, dates, or any other attribute that links with time you can look to techniques for time series analysis or consider options for viewing or processing continuous data. Where you are interested in behaviour or values over a period of time then the popular visualisation approach will be a line chart though we'll talk about this in more detail in a later course in this program.

Geographic (or Spatial): identifying a dataset attribute that has place names, GPS locations or any other spatial description can indicate that visualisation using a map, floorplan or graph is possible. Calculating the distance between spatial positions or plotting the route or trajectory may help to reveal extra information.

Documents: we will talk more about data and document formats later but if you have a data set that is a collection of structured or unstructured documents then you will likely need to do some further processing before the data can be used. This may include natural language processing or text analytics methods.

Multimedia: technically, multimedia are documents that include two or more types (multi) of media (image, audio, video, 3D, etc.). To use the knowledge in multimedia requires content-based or semantic media analytics methods including computer vision, speech-to-text and artificial intelligence techniques to turn the signal (e.g., audio, pixels, motion) into meaningful labels (bird, laughter, walking, etc.). This is a specific and very exciting area of research (coincidentally this is my area!).

References

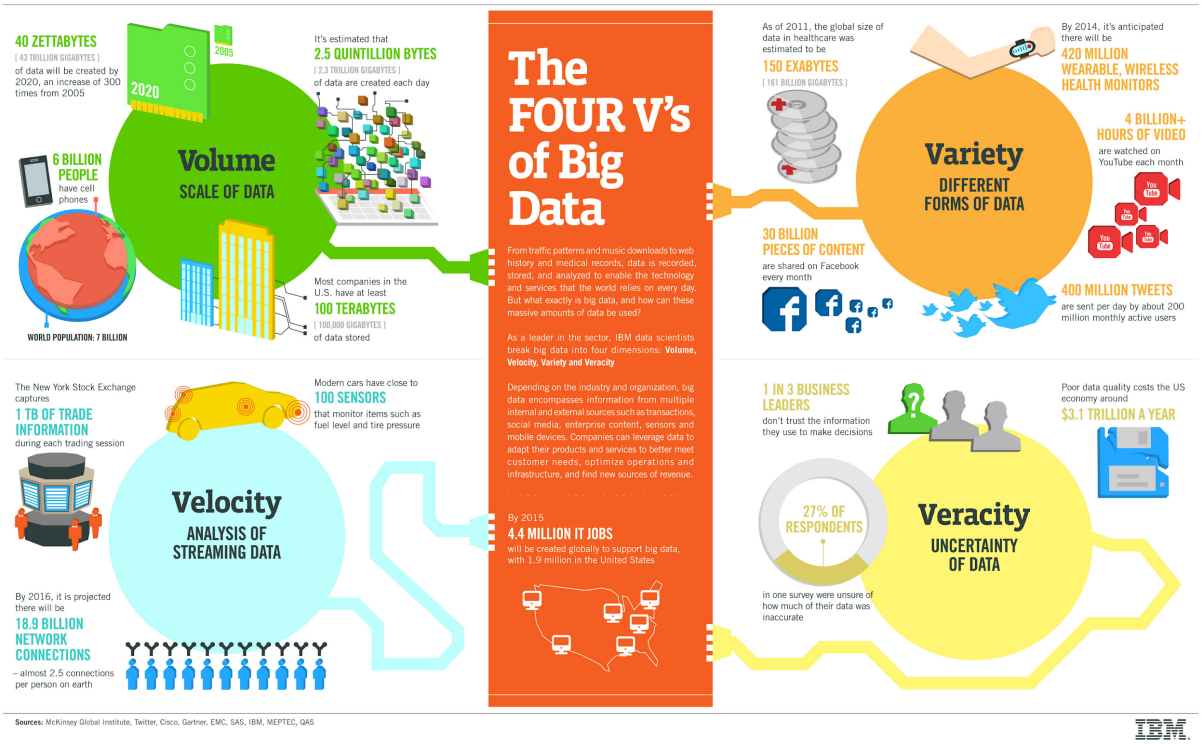
“Types of Data”: Chapter 2 of “Principles of Data Science”, Sinan Ozdemir (2016), https://capitadiscovery.co.uk/dcu/items/931090 (DCU Library)

Good summary of NOIR for data measurements: https://www.questionpro.com/blog/nominal-ordinal-interval-ratio

Nominal, ordinal, interval, & ratio data (simple reference document) http://www.psy.gla.ac.uk/~steve/best/ordinal.html

Big Data

Suzanne Little, suzanne.little@dcu.ie



http://www.ibmbigdatahub.com/infographic/four-vs-big-data

Review the slides and video to see what qualifies as "big" data and to learn about the 3Vs - Volume, Variety and Velocity (optionally including Veracity!). Here we look at some sources and consequences of big data.

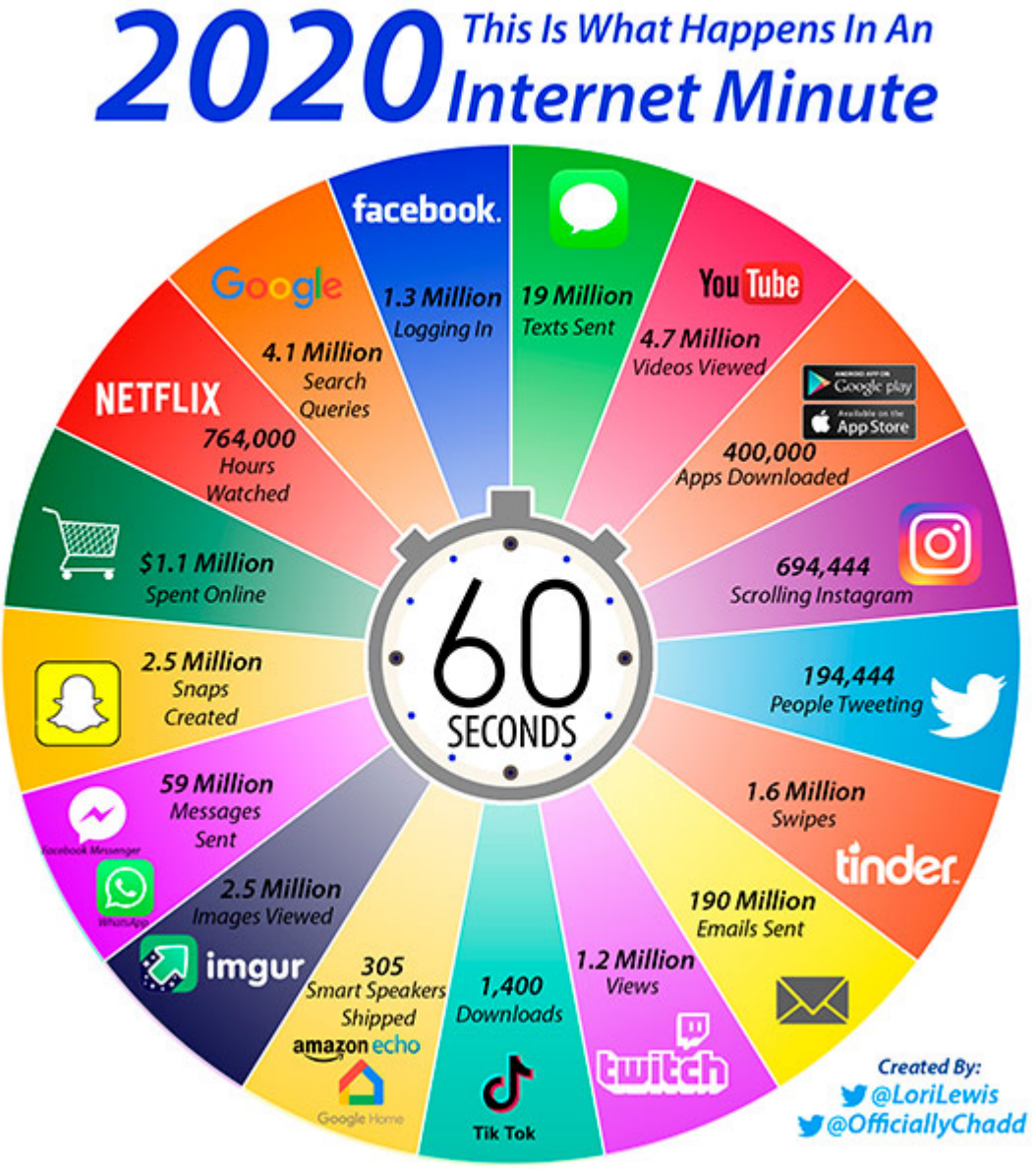
According to Subrahmanyam, (2008):

> *"First, there is human-generated data via social media, and on various online forums, which can be analyzed using standard tools for text analysis. Second, there is process generated data, via the act of purchases and sales, such as credit card receipts, supermarket scanners and so on. Finally, there is machine-generated data, such as tracking GPS movements of delivery trucks, and satellite-based images of parking lots."*

Here are three examples of activities or application domains that generate big data. Can you think of others?

1. The Internet Minute

Internet activity whether eCommerce, entertainment or social media is a rich source of data that is high volume, high velocity and high variety. The graphic below shows some of the activity that occurs each minute (24 hours a day!) in 2020. Can you find some of the graphics from earlier years and see what services have changed?



What happens in an Internet minute [Source]

2. Instrumented Vehicles

YouTube has 500 hours of video uploaded per minute according to Tubefilter. This ranges from cat videos to twitch streams to tutorials to original content, trailers and music videos. This is a clear example of “big data”, high volume (new content is added 24/7), high velocity (that’s over 3 **years** worth of video each hour!) and high variety.

A source of big data that you may not have considered is the transport you use. Instrumented vehicles are used to develop advanced driver assistance and safety systems (Zhu et al, 2019). These cars have 6-10 cameras, radar and/or lidar sensors, accelerometers, pressure sensors and other engine sensors. When a car is sent out to record a scenario (such as a highway driving at varying light levels) then high quality video is recorded and between 10-50TB of data is captured per vehicle per day of operation (that is an eight hour window). This is normally high-definition video that then needs to be annotated in exhaustive detail.

Instrumented vehicle as part of the EU H2020 VI-DAS project, [Source]

Similarly a 2016 survey predicted a massive increase in the amount of data generated from planes and some estimates say that a single jet engine can produce 30TB of data from a 30 minute flight.

Data from transportation is clearly high volume, high velocity and also high variety. It can include numeric data from sensors, video, audio communications and other transaction data.

3. The NY Stock Exchange

Financial transactions, such as those on the New York Stock Exchange, where on average between one and six billion shares are traded daily, are also a common source of big data. The transaction data, volumes, prices, and timings all form a very large dataset. In addition researchers have tried to link share prices or activity with newspaper articles (Tetlock, 2007), tweets (Si et al, 2013), noise on the trading floor (Cocal and Shumway, 2001) and emotional levels in conference calls (Mayew and Venkatachalam, 2012).

This data is particularly high velocity as trades occur continuously but also very high volume, and when combined with outside sources high variety as well.

Consequences?

Now that we've established that big data has high volume, high velocity and high variety, what does this mean for our ability to process big data and why would big data be useful?

A simple method for evaluating if you are dealing with a big data set is to try and open the data using a personal computer system, say in a spreadsheet programme or even a text editor. If your system freezes with an error, crashes or is otherwise very slow to respond then you've likely reached the working memory limits of your programme. High volume data is generally **difficult or impossible to open in the memory available on a PC**.

The high velocity of big data is usually seen in streaming data. That is, new data is constantly being generated and added to the current pile of information to be processed. Therefore programmes for reading or analysing the data need to be able to handle a dynamic data stack and should process the new information **fast enough to be able to use the conclusions**.

Finally, the variety of big data creates some practical challenges. The analyst needs to consider **how to process different data formats** (trades, tweets, images, audio recordings, lidar), how to extract the relevant descriptions from this data (often called feature engineering) and then choose appropriate machine learning models that can handle multivariate data sets.

There are ways to work around the data limits of standard programmes. Working in batches, filtering unneeded data entries using a different application or purchasing more memory are all effective workarounds. In addition, there are specific tools that aim to handle big data and we'll look at some of these later in the course.

But what about the real-world consequences of big data?

Read this article that provides a counter-narrative to the "promise of big data" articles and compare with statements from this overview of a 2013 book on big data.

Like many aspects of technology (AI, Machine Learning, genetic testing, e-finance) there are positive and negative consequences to how we choose to use it. Consider the following statements about big data:

1. Big data allows us to make more personalised decisions like individualised health care and other customised solutions.
2. Big data lets us optimise many traditional processes to provide better services, more efficiently and with greater benefits.
3. “Privacy is harder to protect because the traditional legal and technical mechanisms don’t work well with big data.”
4. Big data is making us safer, smarter and better prepared for the future.

Discuss: Which statements do you agree with? What are your concerns about how big data might be used?

References and further reading

Coval, J.D. and Shumway, T., 2001. Is sound just noise?. *The Journal of Finance*, 56(5), pp.1887-1910.

Insights, S. and Insights, B., 2020. *Big Data: What It Is And Why It Matters*. [online] Sas.com. Available here.

Lee, C.H. and Yoon, H.J., 2017. Medical big data: promise and challenges. *Kidney research and clinical practice*, 36 (1), p.3.

Mayew, W.J. and Venkatachalam, M., 2012. The power of voice: Managerial affective states and future firm performance. *The Journal of Finance*, 67(1), pp.1-43.

Si, J., Mukherjee, A., Liu, B., Li, Q., Li, H. and Deng, X., 2013, August. Exploiting topic based twitter sentiment for stock prediction. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics* (Volume 2: Short Papers) (pp. 24-29).

Subrahmanyam, A., 2019. Big data in finance: Evidence and challenges *Borsa Istanbul Review*, *19*(4), pp.283-287. Available here.

Tetlock, P.C., 2007. Giving content to investor sentiment: The role of media in the stock market. *The Journal of finance*, 62(3), pp.1139-1168.

Zhu, L., Yu, F.R., Wang, Y., Ning, B. and Tang, T., 2018. Big data analytics in intelligent transportation systems: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 20(1), pp.383-398.

Data Quality: Definition, Causes & Measuring

Suzanne Little, suzanne.little@dcu.ie

Here I summarise the main points related to data quality. How to define quality data, the difficulties in measuring data quality and provide examples of causes of data problems from various points in the generic data analytics pipeline.

What is “data quality”?

High quality data is free from both errors and artefacts. An **error** is data that is missing or lost due to the capture process and cannot be recovered. A sensor failure, hard-disk write error, lack of resolution in the image or a street that wasn’t surveyed are all examples of errors. An **artefact** is something that has been introduced into the dataset during the gathering, processing, integration or cleaning activities. If you have good data practises then you can generally identify and address artefacts.

You can also consider whether the poor data is due to **individual** (“one off”) or **collective** (systemic) issues (Piegorsch, 2015). Individual errors occur in a database or dataset that is otherwise of good quality and can include both errors or artefacts. As we will see these are difficult to fully prevent. Collective issues are perhaps more serious and include problems with how the dataset was generated, collected or sampled.

Data quality issues can be compounded by “big” data. Recall that big data is characterised by Volume, Variety, Velocity and Veracity. Each of these increase the complexity and consequences of data errors.

- Variety: diversity of data sources brings abundant data types and complex data structures and increases the difficulty of data integration.
- Volume: significant time is required to fully explore and understand the data.
- Velocity: The data changes quickly and the “timeliness” of data is very short so judgements on consistency need to happen rapidly.
- Veracity is a measure of data quality and the other ‘Vs’ combine to decrease consistency and accuracy of very large, complex data sets.

For further discussion on data quality and big data see (Cai & Zu, 2015).

Consequences of poor quality data

Before we go any further let's first consider why data quality and data cleaning is so important.

A summary article, that quotes from IBM and Experian studies, has estimated costs of $3.1 trillion per year for the US economy and lists a number of other societal and economic consequences.

An often-used example of expensive consequences from a simple data error comes from NASA. In 1999, the $125 million Mars climate orbiter was destroyed when incorrect units used by a contractor and NASA engineers (metric vs imperial) caused it to have the incorrect trajectory. You can read a more complete description of the events available here.

Types of data errors

Part of your data cleaning toolkit is a sense of what might be wrong. The list of examples of formatting and content issues below comes from chapter 4 of The Data Science Handbook by Cady Field.

Formatting Issues:

- Formatting Is Irregular between Different Tables/Columns
- Extra Whitespace (select for "ABC" or "ABC ")
- Irregular Capitalization
- Inconsistent Delimiters (commas or tabs)
- Irregular NULL Format ("" or NULL or NA)
- Invalid Characters (ascii or unicode)
- Weird or Incompatible Datetimes (Day-Month or Month-Day, timezones)
- Operating System Incompatibilities (eg, '\n' or '\r\n')
- Wrong Software Versions

Content Issues:

- Duplicate Entries
- Multiple Entries for a Single Entity
- Missing Entries
- NULLs (what do they mean?)
- Huge Outliers
- Out-of-Date Data
- Artificial Entries (eg, €999,999 salary)
- Irregular Spacings (eg, gaps in time series)
- Incorrect or inconsistent units (metres or inches, dollars or euros))

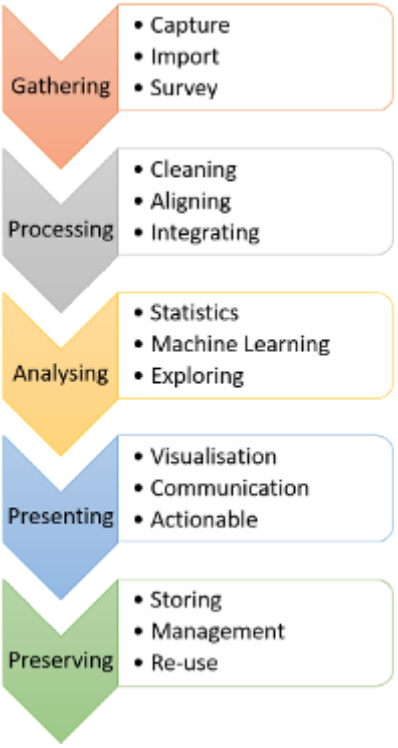
For more examples of bad data, take a look at this article. Do you recognise some of the types of data errors that are described?

To build up your tool kit for data cleaning it's helpful to have a set of core questions and actions that you start a project with. This may include:

- Checking the source file by looking at the raw text (if available) and understanding the format used.
- Reviewing any available documentation and utilising any contacts who may have worked on the data.
- Build up your set of core questions. For example, I'm always suspicious of datetime values; currency; NULL values; strings/integers/floats and matching the count of entries between files or tables.
- Implement sanity checks throughout the data ingestion and processing stages. View the first few rows of the dataset once you've read it in, check that number of rows and columns and that they match your expected values, look for outliers, etc.

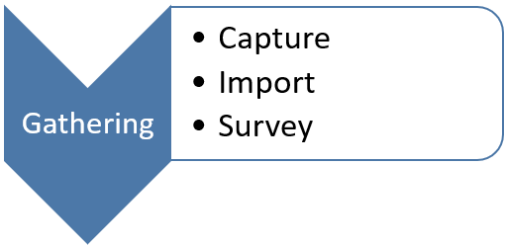
You can read an example of someone's description of their data cleaning process here.

Below is the generic data analytics pipeline that was introduced at the start of this course. There are many places in this pipeline where problems can occur but especially in data **gathering**, **integration** (processing), **analysis** and **storage**.



Data Errors in the Gathering Phase

A common and obvious stage for errors and artefacts to creep into your data occurs during the data-gathering phase.



Recall that errors are generally missing data that cannot be recovered. Clearly there are many opportunities for signal loss or other transmission problems to cause errors. Here we will look at example issues in the data delivery, capture or gathering activities and some actions to address or prevent these. Can you think of any others?

Example errors or artefacts

1. **Data delivery issues** such as transmission problems that may result in loss of network connectivity, buffer overflows or corruption. Other examples of data delivery issues may include inappropriate aggregation of data or format conversions especially of NULL or missing values.
2. **Manual entry errors or artefacts** caused by typos, lazy people etc…. (e.g. DUC instead of DCU) or duplicate entry.
3. **Poor survey or interface design**: e.g. what colour is your toothbrush? Where the only options are red, green or blue! What if it's multicoloured? Another example would be a drop-down for entering your age that only goes back to 1920 (No one alive older than 100?)
4. **No standards** for format or controlled vocabulary for fields, e.g. DCU or Dublin City University? Is it centimeters or inches?

Reflect: Can you think of any examples of things that might go wrong when gathering data? Have you experienced any yourself?

Solutions

There are two types of solutions for issues in the data-gathering phase.

Preemptive: build in integrity checks and entry constraints. This is where you try to imagine what could go wrong and implement either constraints in your system or put process management actions in place to enforce and/or reward accurate data entry. For example, having critical data entry checked by other parties or using data type constraints in your entry form.

Retrospective: focus on cleaning through, for example, duplicate removal, merge/purge, name & address matching or field value standardization. This approach has a diagnostic focus to try and automate the detection and repair of artefacts, errors and glitches.

Often both approaches will be required for critical systems though if you aren't in charge of the data gathering process then implementing and enforcing preemptive checks can be difficult.

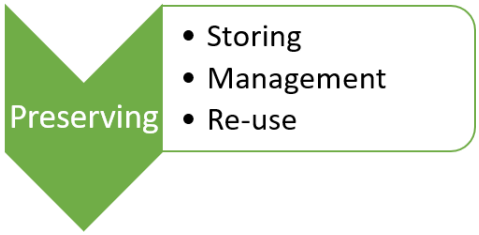
A key part of preemptive strategies is to avoid data loss during transmission. A few specific actions to consider here are:

- Build reliable transmission protocols and use appropriate server infrastructure (eg. redundancy, relay servers, backups)
- Verification such as the use of checksums for downloaded data or implementing a verification parsing step (eg. do the provided files fit an expected pattern?)
- Using interface and provider agreements where a third party is involved. This may involve a contractual data quality commitment from the data stream provided. Consider though, how could this be measured and enforced?

Reflect: preemptive solutions are important and useful but think about the examples from poor survey design. These are caused by the implementation of constraints in the user interface but result in data errors and artefacts. How would you go about avoiding this problem?

Data Errors in the Storage Phase

While data preservation and storage comes at the end of our pipeline, we will look at how this can cause data errors and artefacts here as it is often connected with data gathering issues.



Example errors or artefacts

1. **Format conversion errors** that can happen silently when importing or storing data such as string or float or date (1918 or 2018?). Perhaps you remember the Y2K bug! Format conversion errors may also include rounding or approximation such as a database field limited to integers.
2. **No metadata** or schema information recorded, e.g., What does the field mean?
3. **Poor process** and methodologies: for example, there is no filename convention or version control.

It's also possible to have technical issues such as physical disk failure or corruption and transmission errors such as network dropout.

Reflect: Can you think of any examples of things that might go wrong when storing or saving data? Have you experienced any yourself?

Solutions

1. Document and publish -- remember, good metadata!
2. Data exploration and retrospective checking -- this is sometimes called the “smell test”. Does anything in the data look funny, odd or unexpected?
3. Assume that the worst might happen! And try to have contingency plans.

Reflect: Good solutions to avoid data storage issues are mostly preemptive and require you to have a good imagination! Consider an example storage error or artefact that you thought of earlier, how would you go about avoiding this problem?

Data Errors in the Processing Phase

While this is the phase where cleaning happens, the processing phase includes integrating data from multiple sources and this is a key opportunity for errors or artefacts to creep in. Datasets might be combined because of acquisitions, across departments or organisations or perhaps your problem requires you to bring disparate data together. The cleaning process itself can also introduce errors so you should always work on a copy of your original data!

Example errors or artefacts

1. Heterogeneous data: no common key, different field formats, approximate or exact string matching.
2. Different definitions: What is a customer, an account, a sale, a book, an address etc...
3. Time synchronization: Does the data relate to the same time periods? Are the time windows compatible? Do you understand the time format (American month-day-year?).
4. Legacy data in multiple formats: database systems, spreadsheets, ad-hoc files, binary data, and many other formats.
5. Sociological factors: knowledge is power so the person you are requesting the data from may be reluctant to share or help.

Reflect: Can you think of any examples of things that might go wrong when joining two data sources? Have you experienced any yourself?

Solutions

1. Commercial Tools (e.g., from IBM, Oracle, SAP, etc...): These are the result of a significant body of research in data integration. Many tools are available that perform convenient and powerful tasks such as address matching or schema mapping.
2. Data browsing and exploration: You can use many other tools to browse the metadata -- either provided or extracted by you. By viewing the before and after

results (number of entries, expected data types, etc.) you can check if the integration happened as expected.

Data integration is a common task and there are many tools and processes available. It is often required when two companies merge and need to join their IT or customer management systems. Like many of the possible errors, the best defence is a good idea of what the end result should look like and the imagination to think of what may go wrong. This will enable you to to prevent or check for errors.

Data Errors in the Analytics Phase

While we only really cover data exploration in-depth in this module, the activities involved in data mining and analysis can also introduce artefacts into a data set. This is especially true where data is being summarised or features consolidated for machine learning.

Example sources of errors

It's helpful to stop and think before plunging into the analysis and aggregation of data. What are you planning to do with all this data anyway? This will help you to understand the level of precision required, the types of features, the importance of time-series granularity (data from every second or every minute) and what the needs are of the analytics model you are hoping to create.

Some common problems in analysis are:

- Scale and performance: how much data are you processing? Will you run into memory or processing issues?
- Confidence bounds? 0.95, 0.99? Is 99.9999% accuracy required? What are the real-world consequences of getting incorrect predictions?
- Attachment to models: while it's true that bad data trumps a good model, don't always assume there is no problem with your model.
- Insufficient domain expertise: the analyst may not know a specific fact about the real world that explains a trend or outlier.
- Casual empiricism (use of an arbitrary number to support a pre-conception). You may not know that numbers in your data are casual or you may be guilty of this yourself. 85% of all statistics are made up on the spot! (note: this is a made up number).

While it is good advice to try and consider what your data or model will look like before you integrate and analyse, don't forget the quote from Sherlock Holmes.

> “I never guess. It is a capital mistake to theorize before one has data. Insensibly one begins to twist facts to suit theories, instead of theories to suit facts."
> [Arthur Conan Doyle, Sherlock Holmes in A Scandal in Bohemia]

Reflect: Can you think of any examples of things that might go wrong when analysing data? What do you think is the cause?

Solutions
Engage the smell test! This is the most important skill you can develop; a finely-honed sense of potential problems and an awareness of what your data is telling you.

Generally, you want to build the following tasks into your data processing and analytics workflow.

1. **Data exploration**: determine which models and techniques are appropriate, find data bugs and develop domain expertise.
2. **Continuous analysis**: Are the results stable? How do they change?
3. **Accountability**: Make the analysis part of the feedback loop.

Reflect: You have now seen many examples of problems that can occur during different phases of a data-driven project. What do you think is the most useful action you can take to avoid getting or using “bad data”?

Measuring Data Quality

So we’ve now seen that poor quality (“bad”) data can cause big problems and there are many ways that you can get errors or artefacts into your data. But how can you measure the quality or messiness of your data?

If I ask you to come up with some ways of measuring or assessing the quality of your data, perhaps you might suggest ideas like the following:

- Accuracy: the data was recorded correctly.
- Completeness: all relevant data was recorded.
- Uniqueness: entities are recorded once.
- Timeliness: the data is recent or kept up to date (date.updated).
- Consistency: the data agrees with itself (internally consistent).
- Credibility: the data comes from a recognised (or official) source.

All of these are possibly useful ways to think about data quality but there are problems with using these as formal measures.

Firstly, many of them are unmeasurable. It is very difficult, perhaps impossible to measure accuracy and completeness. You can’t be sure of what you don’t know!

Secondly, the context in which you are gathering and using the data is important and these measures are not content-independent. For example, if you are computing aggregate values from the data then you can tolerate more inaccuracy than if you are measuring an outcome more precisely.

Thirdly, you probably thought of other measures that would be useful such as interpretability, accessibility, metadata and analysis. All of these may also be important measures of data quality depending on your context.

Finally, the measures themselves are vague and difficult to define and quantify.

Maybe you also thought of some practical ways to ensure data quality and therefore to measure the accuracy of your data. To prevent errors (data loss) you could use checksums or have static schema *constraints* (e.g., no null values) or you could introduce data processing protocols (e.g., all international orders are processed by Accountant A). You could keep adding more and more constraints to try and detect or prevent possible data errors. Of course, this is very difficult and it's almost impossible to know when you've added enough constraints to never have a data error. You also increase the likelihood of added data artefacts.

However, with *static* and *dynamic* constraints you can measure things like adherence to the defined schema or adherence to the business process and get a measure of potential data quality.

Other practical means to judge data quality include:

- Inventory (expensive)
- Using a proxy measure such as tracking customer complaints
- Applying formal measures of accessibility
- Using test cases with known results and checking for glitches or errors in analysis
- Successfully completing an end-to-end process (e.g., data ingestion, processing, indexing, querying and summarisation)

If you have studied software engineering you may recognise some of these principles from software testing and, just like testing complex software, it is very difficult to be sure that you have found all the bugs.

So, if it is so difficult to define and quantify data quality but we know that high-quality data is needed to produce good results, what do we do? Data cleaning is a little like being an art restorer. **You need a variety of skills, experience of different circumstances, general knowledge, a well-worn toolkit and some intuition** and you need to try and avoid making more problems (introducing more artefacts or errors). Start building your toolkit by looking at the Exercises using Spreadsheets, OpenRefine and Python/Pandas.

References

Cady, F., 2017. 'Chapter 4: Data Munging: String Manipulation, Regular Expressions, and Data Cleaning'. *The data science handbook*. John Wiley & Sons DCU Library eBook: https://doi-org.dcu.idm.oclc.org/10.1002/9781119092919.ch4

Cai, L., & Zhu, Y. (2015). The Challenges of Data Quality and Data Quality Assessment in the Big Data Era. Data Science Journal, 14, 2. DOI: http://doi.org/10.5334/dsj-2015-002

Bertil Hatt, What does bad data look like?

Piegorsch, Walter W.. *Statistical Data Analytics : Foundations for Data Mining, Informatics, and Knowledge Discovery, Solutions Manual*, John Wiley & Sons, Incorporated, 2015. ProQuest Ebook Central, http://ebookcentral.proquest.com/lib/dcu/detail.action?docID=1895958.

Online Course on Data Cleaning: https://www.europeandataportal.eu/elearning/en/module11

Data Cleaning 101: https://towardsdatascience.com/data-cleaning-101-948d22a92e4

Elite Data Science, mini-course on Data Cleaning: https://elitedatascience.com/data-cleaning

Data Cleaning: Tools and Tips

Tools for Data Cleaning

There are many possible tools for cleaning data including general purpose systems and specific platforms.

Remember that the purpose of data cleaning is to identify errors and artefacts and fix them where possible. Good data cleaning should be well documented so that others can identify any problems that could impact on statistics, analytics, machine learning and conclusions reached from the data.

Here I've listed some resources and the strengths/weaknesses of each tool.

	Strengths	**Weaknesses**	**Resources**
Spreadsheets: Microsoft Excel, Google Sheets, Open Office.	widely available, relatively easy to use interface, support for many common data file formats, some advanced tools for data reconciliation and cleaning	memory limits (desktop or cloud), no audit trail and limited data version management options, lack of repeatability for multi-step cleaning, lack of automation	Cleaning Data with Excel and Google Sheets Top ten ways to clean your data using Excel Using a spreadsheet to clean up a dataset
OpenRefine	easier for non-programmers to use, interactive and graphical viewing of transformations, advanced options for data reconciliation (eg, geographical), auditable changes (complete provenance/undo history of all modifications), wide variety of input/output formats	standalone tool less widely available, yet another tool to learn (if you are already an experienced R/Python programmer), difficult to perform editing actions like splitting rows or inserting new rows (OpenRefine is not a spreadsheet).	Using OpenRefine (Book) Cleaning Data with OpenRefine
Python: Pandas, Numpy	powerful and flexible general purpose language, supports very wide range of input/output sources and formats,	not useful for non-programmers, scripting and has no graphical user interface, can be difficult to	Pythonic Data Cleaning With Pandas and NumPy – Real Python

	if documented and used properly then scripts are auditable and repeatable, cleaning process can be deployed to other parts of the workflow, Regular expressions are extremely powerful	explore or view data without using extra libraries and functionality, Regular expressions require skill and practise to use well	

Other tools (sometimes referred to as “data preparation” functionality)

- Rapidminer: https://rapidminer.com/glossary/data-preparation/
- Tableau Prep: https://www.tableau.com/products/prep

Tips on Data Cleaning

Data cleaning is an important skill and building your toolkit and experience in spotting potential problems will be highly useful. Here let’s recall some of the key steps for effective data cleaning and a starter list of initial checks to make.

First, you should work on a copy of the data where possible. If you are making changes to the content of the data you should be able to revert these changes and it is very easy to accidentally overwrite or delete data without intending to. Keeping an auditable record of the changes you’ve made and being able to reverse them is critical.

Second, be careful not to inadvertently make changes because you don’t understand the domain of the data. Consult with domain experts or the original source of the data where you can. This is useful where common conventions may explain strange outliers or acronyms present in the data.

Finally, be transparent about your cleaning. Annotate the dataset where necessary to indicate where missing values have been imputed or where data has been lost. Consider what the consequences might be of any changes you make -- does it change descriptive statistics or potentially alter a training dataset?

A side comment is that you may have heard that “great programmers are lazy programmers”. This just means that rather than performing a repetitive task they look for ways to automate it or do it more efficiently. Consider how you can become a more efficient data wrangler (ie, cleaning, reformatting, manipulating). Learning how to become proficient in at least one of the tools presented here will be a helpful skillset to acquire.

A few suggestions for initial data quality checks

Here is a list of suggested initial checks. You should start to build your own set of questions and common checks as you explore and clean more datasets.

- Look at the raw data file using a text editor or a function in Linux like top/tail.
- Check the expected data types -- string, float/integer, datetime?
- Date and timestamps are especially prone to error. Format of dates - DD/MM/YYYY, MM/DD/YYYY, Excel based or Unix based, are timezones an issue?
- Missing values, records or variables -- are empty cells no value (0) or no measurement (null)? How should they be handled?
- Erroneous values -- typos or values that are clearly out of place (eg, gender value in age column)
- Inconsistencies -- capitalisation, units of measurement
- Leading or trailing spaces! Windows or Linux end of line characters
- Check for duplicate records but be cautious about deleting without understanding the data
- Out of date records -- e.g., age will have changed
- Perform regular “Sanity checks” - look for extreme values or outliers, count how many records are present.

The best way to become an experienced data cleaner is to practise!

What is a graph or chart?

It's helpful to know the correct terminology to describe the components that make up a graph or chart.

Why? It makes it easier to search for and find documentation and to properly identify problems with a visualisation. Different programs and libraries may use slightly different terms but here is a reference that covers most of the major options you will come across.

The figure below comes from a handout by Stephanie Evergreen. It shows the same data graphed in two different ways and labels the main components.

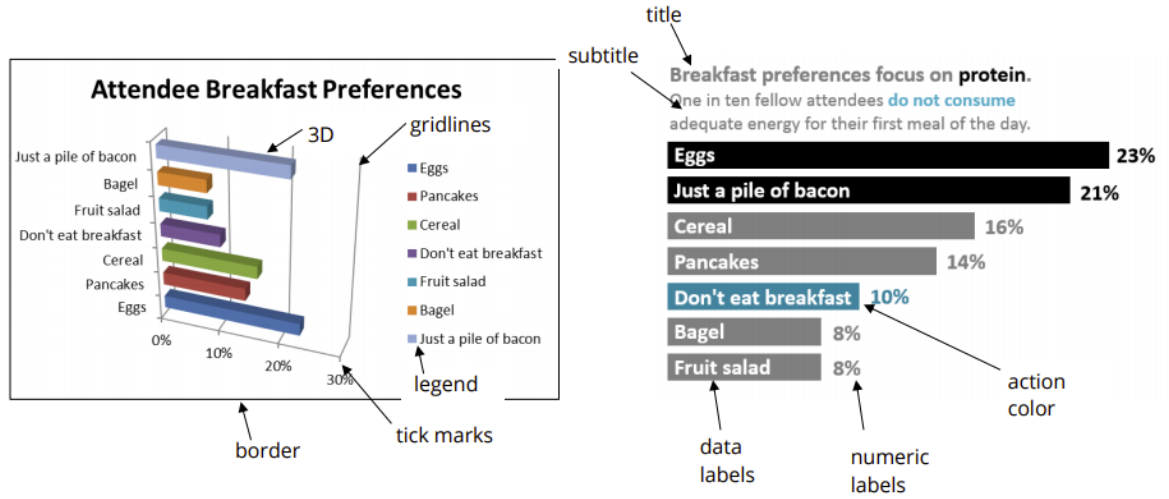


Fig: example graphs with labelled components (from:http://stephanieevergreen.com/updated-data-visualization-checklist/)

Can you find each of the following in the figure above: 3D, gridlines, border, tick marks, legend, title, subtitle, action color/colour, data labels, numeric labels? Notice that the x- and y- axis are not explicitly labelled here. Do they exist in the second graph?

We can also look at how graphs are defined in libraries such as **Matplotlib (Python), ggplot2 (R) and Latex**. The image below labels a figure with the terms used in these libraries. This is especially useful when looking at the documentation to try and change a characteristic (colour, position, value, etc.) for one of these components. Most of these terms are the same for other libraries in python or R.

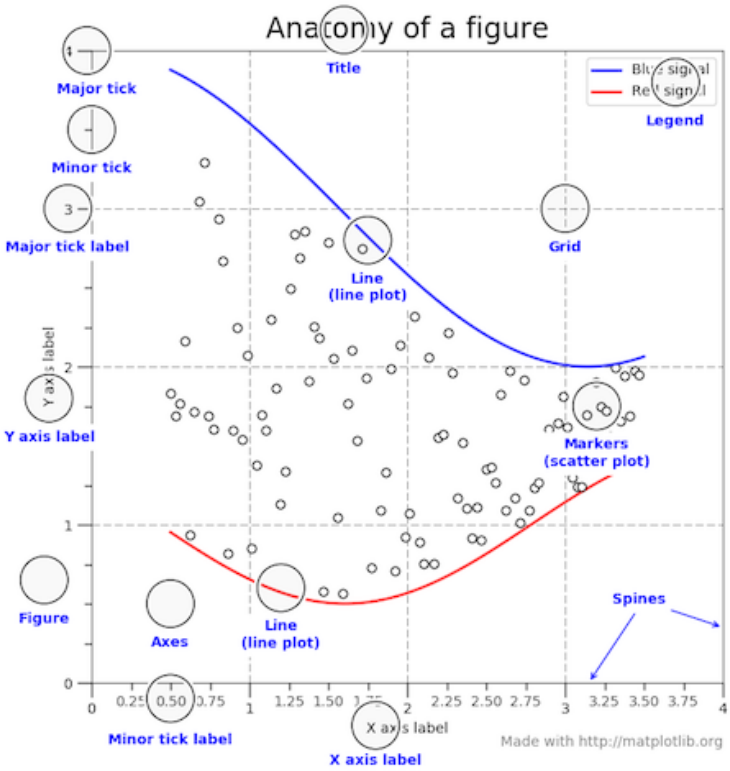


Fig: a labelled figure from Matplotlib showing the names of the graph components (from: https://matplotlib.org/examples/showcase/anatomy.html)

Alert! One very important thing to watch for in matplotlib and related libraries is the difference between an ax**e**s and an ax**i**s. The axes is the area that your plot appears in, not the plural of axis (as commonly used in other documentation). You can have multiple axes in a figure to get a multi-graph figure that may share the same axis/axes. Yes, this is confusing! See the figure below.

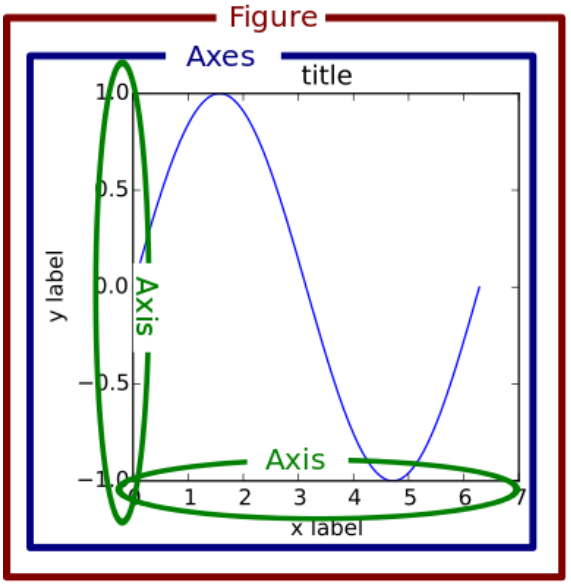


Fig: labelled figure showing the axis and axes in matplotlib (from: https://matplotlib.org/1.5.1/faq/usage_faq.html#parts-of-a-figure)

Google Sheets and Excel use mostly the same terms for “charts” though note the following:
Y-axis = Vertical axis
X-axis = Horizontal axis
Series = sets of related data. A chart can have one or more series. Each chart type displays series differently. Often (but not always), series correspond to rows of data in the data range. Making changes to the attributes for a Series (say the colour) will change the value for all data points in that Series.

More information and definitions for Excel chart components:
http://www.mit.edu/~mbarker/formula1/f1help/10-ch-c2.htm

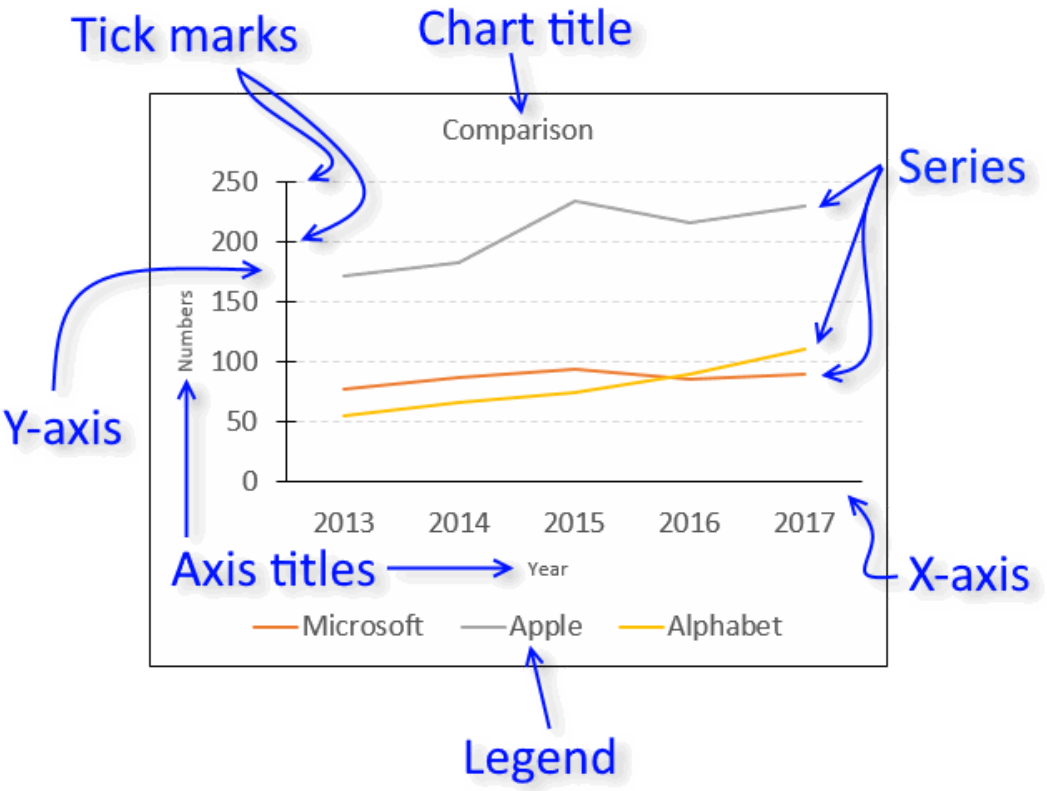
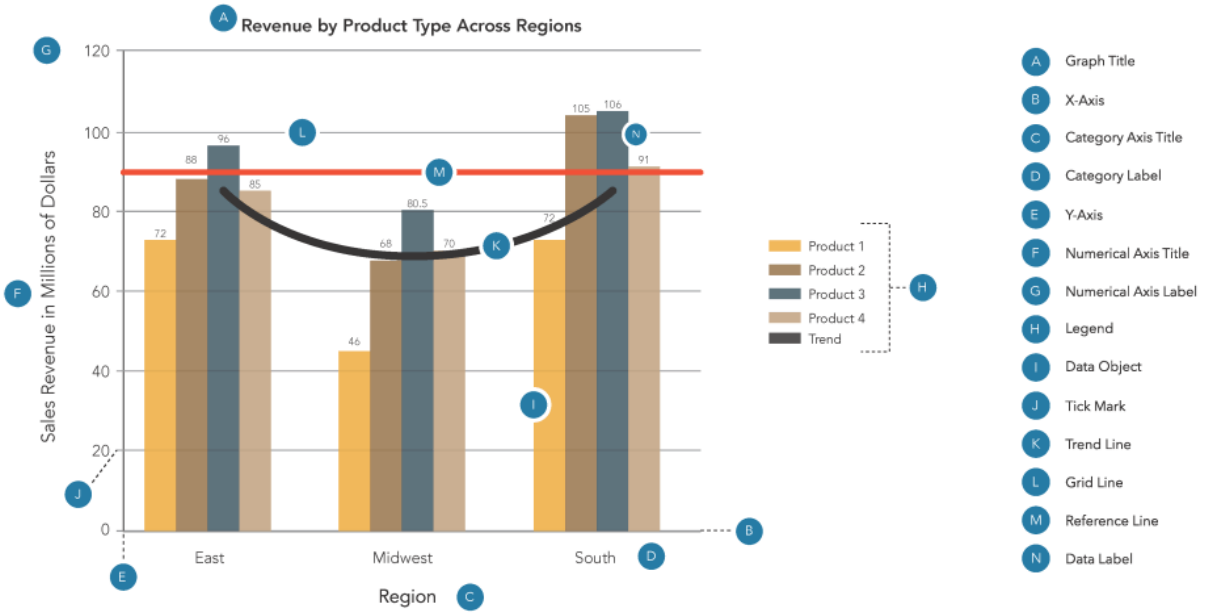


Fig: labelled example of an Excel chart (from: https://www.get-digital-help.com/components-of-a-excel-chart/)



A tool neutral discussion of graph components and some descriptions of each of the elements (https://chaione.com/blog/building-blocks-graphs/).

There are some differences in a declarative language like Vega or Altair or the various graphical applications like Tableau (here's a reference for the terms used in Tableau) but the terms shown in this article should help you get started.

What makes a “good” visualisation?

Clearly the term *good* can’t really be applied to a visualisation (let’s not get philosophical here!) but what we really mean is “**what makes a visualisation successful or effective?”**. That is, the message is clearly communicated through the medium (and the associated design choices) from the sender and understood and acted upon by the receiver (audience). We will return to this again when we look more at:

- How humans see and therefore what impact design choices can have,
- How to encode data (numbers) into visual languages and
- How to gain and lose attention.

For now let’s consult some experts (statisticians, data journalists, visualisation consultants and authors) and see what sort of terms and concepts they say are important for good visual communication starting with a visualisation.

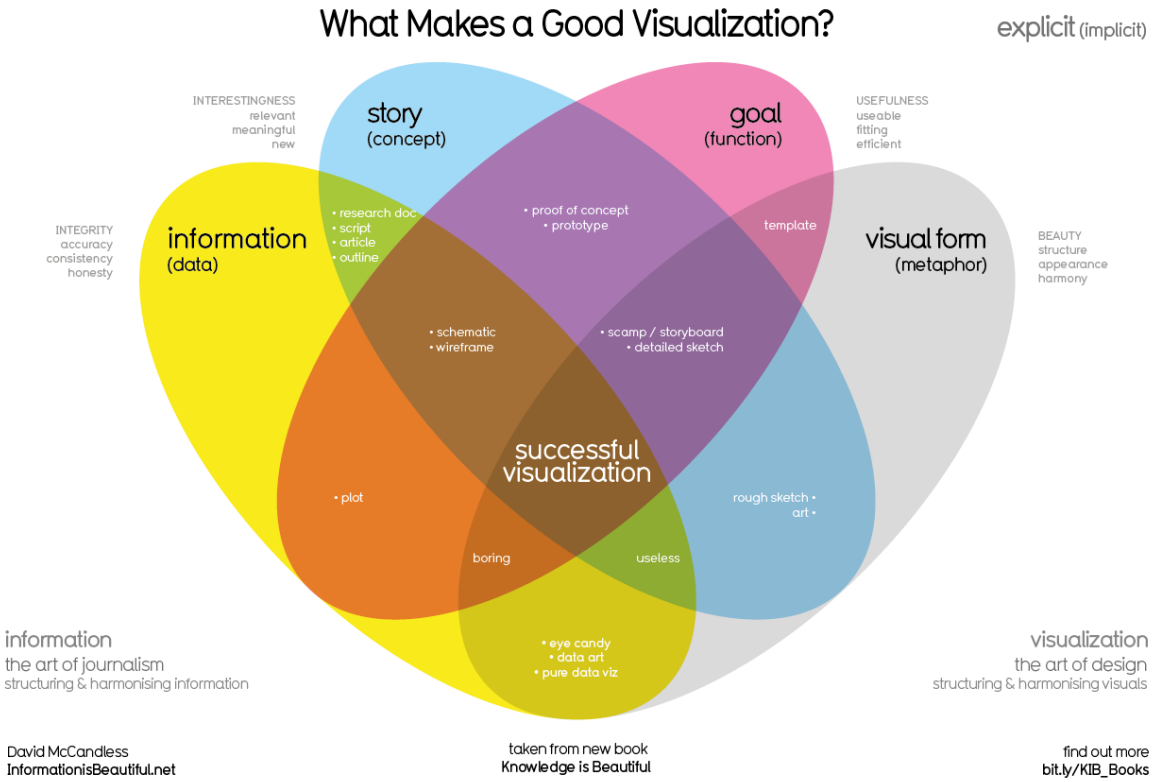


Figure: Information is Beautiful graphic on What Makes a Good Visualization? (David McCandless, 2014).

The image above is a visualisation (specifically a Venn diagram) created by **David McCandless** from his book “Knowledge is Beautiful” (2014) that tries to capture the important aspects (as he views them) of a successful visualisation. He defines these as the intersection of information (data), story (concept), goal (function) and visual form (metaphor). Notice that the intersections that don’t include all options are still useful and valid outcomes (art, template, prototype, etc.) except for two combinations (boring, useless). Can you see which combinations McCandless feels don’t have value? Explore a larger version of this graphic here.

What do other data visualisation experts and practitioners have to say about "good" visualisation?

Edward Tufte: statistician who is widely regarded as a pioneer in the field of information visualisation. Tufte states that the:

> "Viewer should "think about substance rather than methodology …"
>
> "Graphical excellence … gives the viewer the greatest number of ideas in the shortest time with the least ink in the smallest space".

Stephen Few: data visualisation consultant and author

> Well told stories - simple, seamless, informative, true, contextual, familiar, concrete, personal, emotional, actionable, sequential. ("Show Me the Numbers" (2012))

Cole Nussbaumer Knaflic: data visualisation consultant and author, formerly worked in banking, private equity firms and Google.

> "Data visualization is the process of turning information into pictures for a specific purpose." The important components are Process, Information, Pictures and Purpose (2020).

Interview with Knaflic and Andy Cosgreave (Tableau evangelist) giving 3 tips for storytelling with data and Knaflic's response to an earlier version of the above graphic by McCandless.

Stephanie Evergreen: data visualisation consultant and author

> use a "research-based approach to visualization and design".

Alberto Cairo: visual data journalism, read a profile of his work here.

> "Above all else, visualizations — when done right — are a vehicle of clarification and truth."

Gregor Aitsch: former graphics editor of the New York Times, now Co-Founder/CTO at Datawrapper.

> "Know the rules, before you break them …"

Andy Kirk: freelance data visualisation specialist, author of "Data Visualisation" (2016).

According to Kirk (2016) the principles of Good Data Visualisations are:

1. Trustworthy
    a. Don't use inappropriate colour palettes or fonts
    b. Don't include unnecessary chart junk (decoration)
2. Accessible
    a. Useful and understandable
    b. Reward vs Effort (complexity is sometimes okay!)
3. Elegant
    a. Thorough (get this little details right)
    b. Stylish

There's a huge number of resources, opinions, blog posts, interviews, twitter accounts, books, videos, training seminars and more on data visualisation. Even within this small selected group of practitioners there are dissenting opinions on what makes a good data visualisation. This is why it can be tricky to create a definitive and absolute list of what makes a "good" visualisation. Is it the simplicity of Tufte/Few or the data-driven focus of Cairo or the storytelling of Knaflic?

Can you identify a few key principles that you want to remember? Is there anyone else who you are aware of that's doing interesting work in data visualisation?

References

Kirk, A., 2016. *_Data visualisation: A handbook for data driven design_*. Sage.

McCandless, D., 2014. *_Knowledge is beautiful_*. London: William Collins.

Attention, please

Understanding how to direct the viewer's attention is a key part of creating successful visualisations.

Here we look at a metaphor of visual attention called the spotlight and searchlight model which proposes that the focus of attention is analogous to the beam of a spotlight or search light. This limits the higher processing of visual input to those signals within the field of attention. The simplified diagram below shows how the useful field of view is managed by the movement of the eyeball controlled by the ocular muscles.

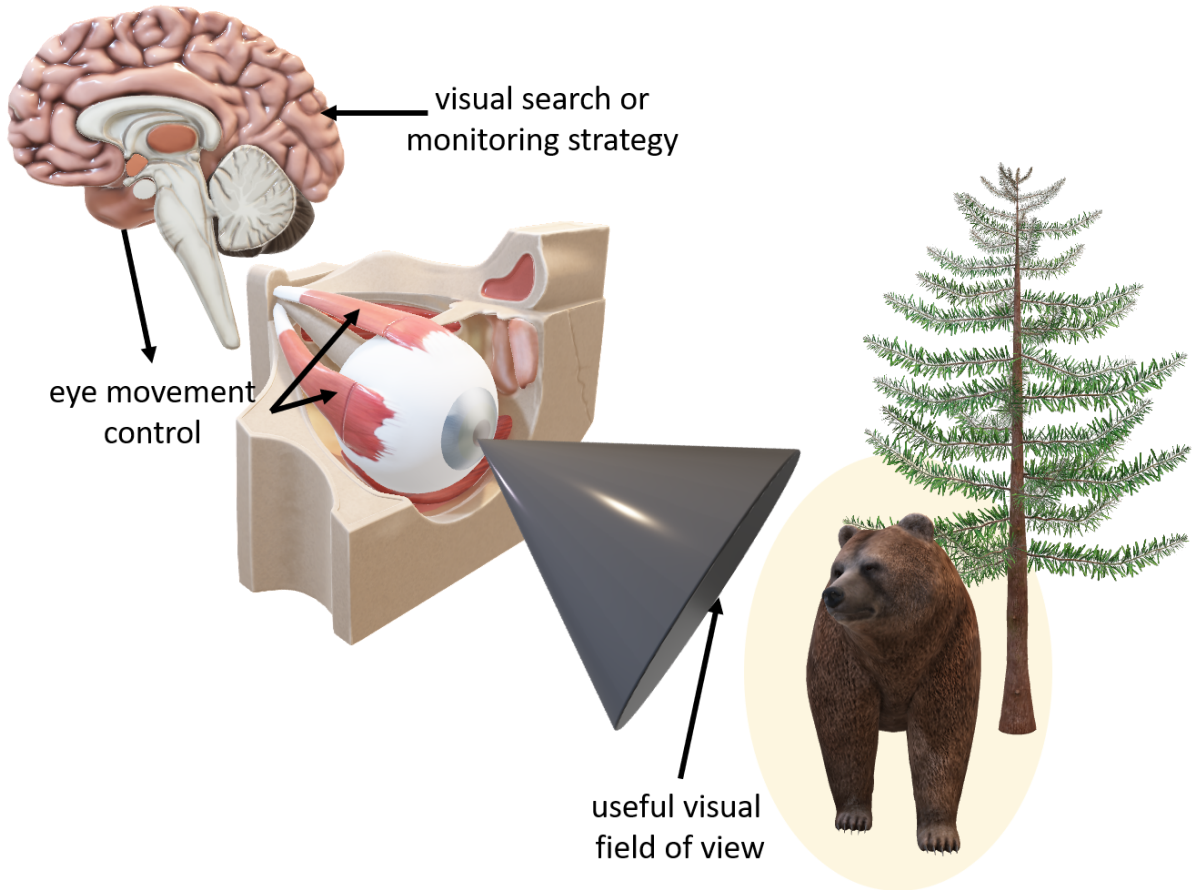


Fig: a simplified diagram (not to scale!) showing the spotlight model of visual attention

The size of the field of view (the spotlight) can vary (a zoom lens metaphor) and this can be influenced by the *density of data* present and the *stress level* of the viewer. A highly dense document might require a very tight focus that moves across the document for the user to interpret the visual input. The stress level is increased with distraction (eg, the brain is also processing audio input or navigating a crowded pavement) and the motivation of the user (a bear would tighten my focus!) based on the need they have to interpret your communication (eg, an emergency evacuation map during an alarm).

Within the field of focus there are operators that we can use to grab attention and we will look at some of those in the next step but you can imagine that movement, flashing lights, bright colours and things that we are interested in (people, our name, money) will grab our attention. These *popout effects* (notice the use of italics!) are general ways of getting attention.

Our eyes move in what is called a series of saccades (rapid movement of the eye between fixed points) so we get updated visual input from shifting the spotlight roughly 3 times per second. But how do we direct this spotlight? This can happen in response to a stimulus or due to scanning of an environment.

One way of measuring this is the concept of “saliency”. This identifies the parts of a scene that the eye tends to focus and linger on first and there are datasets that have been created by putting eye-gaze tracking glasses onto users and overlaying the saliency heatmap to the visual footage (example below). Researchers at the Insight Centre for Data Analytics are working with visual saliency to improve 3D virtual environments and computer vision models.

Fig: Example of visual saliency showing the image on the left and the saliency heatmap on the right where brighter areas indicate the parts of the image we focus on first. (Images provided by Kevin McGuinness (Pan et. al, 2016), based on SalNet data)

Second, there are theories about how we segment a document, dividing up the visual field and guiding attention -- e.g., the Z pattern, Gutenberg and F patterns (diagram below). Designers will use these in advertising, catalogues, web pages, etc. to understand the order that information will be processed in. It’s important to realise that these patterns are theories of what happens where there are limited or no visual hierarchy cues and generally apply for text heavy content like web pages. However, it’s useful to consider the importance of the top-left corner and top of a visualisation in particular (the “golden triangle”). We will look at attentive features in the next section to understand how to use visual attributes (colour, shape, etc.) to direct attention. To learn more about how designers use these patterns see this article.

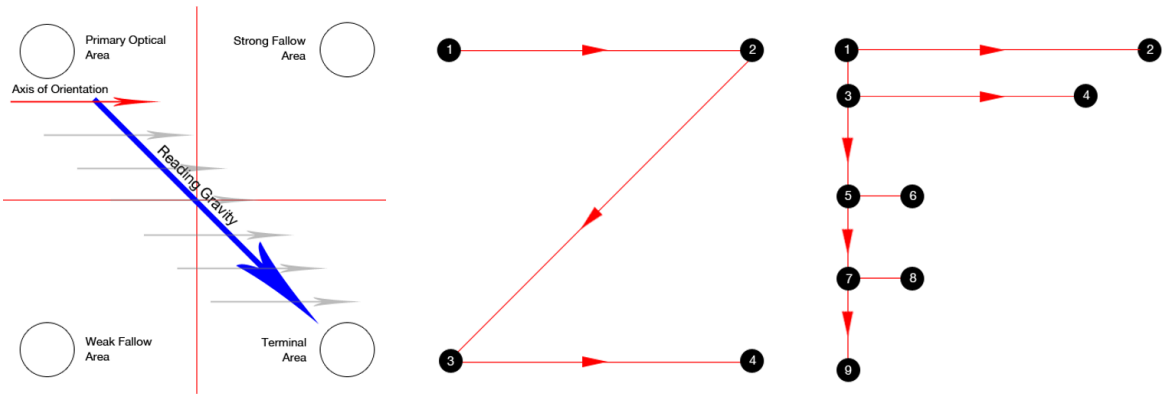


Fig: Gutenberg (L), Z-pattern (M), F-pattern (R) for visual attention (from: https://vanseodesign.com/web-design/3-design-layouts/)

Finally let's try an experiment to see how good your visual attention is. Watch the video below that will ask you to count the number of passes of a basketball the team in white makes.

https://youtu.be/0grANlx7y2E

Now that you have appreciation for visual attention and a simplified model of how it works this will have two benefits for your visual designs:

1. To exploit attention to highlight the data or marks that directly relate to the message you are trying to communicate.
2. To appreciate that attention is fragile and distraction can dilute your message. In particular there are certain visual attributes that are processed faster or differently to others. Look at the document on "preattentive features" to see which visual attributes grab attention.

Data Management Tools

CA682, suzanne.little@dcu.ie

An overview of why we store data and introduction to five general categories of data storage methods.

Let's start by considering why data management tools such as databases are necessary. Why couldn't we just keep all our data in simple, flat text files like CSV? If we have good metadata, as discussed previously, then surely this is enough effort! We need to think about questions of data persistence and access.

Databases and data storage options allow you to:

- **Find** relevant information quickly
- **Select** required data subset for further analysis
- **Apply** analysis functions to (large, distributed?) datasets
- **Calculate** regular reports as data is updated
- Ensure **consistency** and **protection** of data

What makes a database? You can consider the following characteristics to be found in most data storage methods.

1. Structure - tables, documents, "chunks"
2. Minimise redundancy - efficient storage, normalisation
3. Maintain consistency - updates, transactions, deletions
4. Multiple user and concurrent access
5. Query options - eg. language like SQL, SPARQL, Cypher
6. Security

So a database is a means of storing a structured set of data (though the data itself may be unstructured) so that it can be accessed, managed and (easily) updated. The database is not only about the storage but also about how you can use the data.

In this course we consider five general categories of data storage approaches:

1. Relational databases (traditional & modern)
2. Column databases
3. MPP or Data Warehouses
4. No SQL data storage methods
5. Big data methods (e.g., MapReduce, Hadoop/HDFS)

There are a lot of other ways of labelling database management systems but these five give a useful, general grouping for data-driven purposes. There are also new systems being released. For example, https://sno.earth/ is a way of storing geospatial data using the git distributed version control system. Is this a database?

This field is full of acronyms! Here are a few that you might come across - can you think of others?

- CRUD: Create Read Update Delete or the four basic types of SQL commands that illustrate required functionality for data storage options.
- ETL: Extract Transform Load
- DBMS: Data Base Management System
- RDBMS: Relational Data Base Management System (a specific type of DBMS)
- SPARQL: SPARQL Protocol and RDF Query Language (for querying linked data)
- SQL: Structured Query Language, commonly used in RDBMS

In this document we will look at the general principles of each data storage category and some specific examples of tools. As we discuss the different options you should take note of the strengths and weaknesses and the different scenarios when these tools are most appropriate. Database administrators and database experts require a lot more knowledge about systems and engineering but as a data analyst or machine learning expert it is useful for you to be able to understand the issues, specify your requirements and know the vocabulary and acronyms used.

There are many options for comparing or ranking databases. Take a look at the ranking by db-engines.com (screenshot from 2020 below). Notice how many of them are marked as *Multi-model* under the Database Model column? Many traditional relational databases have added functionality to include options like graph, document, time-series and RDF storage so they can be used for a variety of data types. You can also create solutions that use combinations of different tools to form what is known as a "polyglot persistence" solution.

358 systems in ranking, September 2020

Rank			DBMS	Database Model	Score		
Sep 2020	Aug 2020	Sep 2019			Sep 2020	Aug 2020	Sep 2019
1.	1.	1.	Oracle	Relational, Multi-model	1369.36	+14.21	+22.71
2.	2.	2.	MySQL	Relational, Multi-model	1264.25	+2.67	-14.83
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	1062.76	-13.12	-22.30
4.	4.	4.	PostgreSQL	Relational, Multi-model	542.29	+5.52	+60.04
5.	5.	5.	MongoDB	Document, Multi-model	446.48	+2.92	+36.42
6.	6.	6.	IBM Db2	Relational, Multi-model	161.24	-1.21	-10.32
7.	7.	↑ 8.	Redis	Key-value, Multi-model	151.86	-1.02	+9.95
8.	8.	↓ 7.	Elasticsearch	Search engine, Multi-model	150.50	-1.82	+1.23
9.	9.	↑ 11.	SQLite	Relational	126.68	-0.14	+3.31
10.	↑ 11.	10.	Cassandra	Wide column	119.18	-0.66	-4.22

Fig: rankings of databases from (db-engines.com)[https://db-engines.com/en/ranking] (Sept 2nd 2020)

Which of the top ten databases from db-engines.com have you used before?

You can also check out a visualisation (!) of the genealogy of relational database systems.

Relational Databases

Relational DBs: the all-purpose solution for not-that-big data

Relational databases (RDBMS) are the most common type of database and you've probably used them for projects previously. They are structured according to the *relationship* between the data and consist of Tables, Records and Columns. The relationships (links) between these elements facilitate searching, organisation and reporting on the data.

Traditional Relational DBs are:

- familiar and easy to use.
- have comparatively low resource requirements.
- well supported APIs for a big range of software and libraries.
- sophisticated querying through SQL.
- not suitable for really big data, especially if query results are greater than available working memory.

Modern relational DBs and individual implementations often have considerable extra functionality that enables them to support big data or unstructured, non-relational data.

Examples

- **Oracle**: one of the original and longest lived DBMS, very powerful and very popular.
- **PostgreSQL**: open-source, relational database system, used for over 30 years.
- **MySQL**/**MariaDB**: MySQL was purchased by Oracle and forked to produce MariaDB (the open source version maintained by the original MySQL developers).
- **IBM DB2**: now a family of hybrid data management products that includes the original RDBMS.
- **Microsoft SQL Server**: now comes with big data functionality provided by Apache Hadoop and HDFS.
- **SQLite**: very light-weight RDBMS option, "C-language library that implements a small, fast, self-contained, high-reliability, full-featured, SQL database engine." Claims to be the most used database system in the world.

Legacy databases are a common source of data. Retrieving via queries or exporting in a structured format is straightforward but this type of data can be a source of integration issues if the database structures are not well documented. Most programming languages will support APIs (ODBC, JDBC, etc.) that allow for connections to remote database servers that can then run SQL queries to create, read, update and delete entries (CRUD functions).

Column Databases

Columnar stores : rather than indexing rows like a traditional relational database, index by column.

Columnar stores or column databases are a type of database that stores data in a column oriented model. Compare this with relational, which uses a row oriented model and object-oriented databases or document model storage.

Some key benefits of columnar databases include:

- Compression. Column stores are very efficient at data compression and/or partitioning.
- Aggregation queries. Due to their structure, columnar databases perform particularly well with aggregation queries (such as SUM, COUNT, AVG, etc).
- Scalability. Columnar databases are very scalable. They are well suited to massively parallel processing (MPP), which involves having data spread across a large cluster of machines – often thousands of machines.
- Fast to load and query. Columnar stores can be loaded extremely fast. A billion row table could be loaded within a few seconds. You can start querying and analysing almost immediately.

This makes columnar storage a suitable option for big data. They are useful for many business analytics applications where aggregation and transformation of highly structured data is commonly performed. Not generally considered as the best option for data mining and can be slow to add, update or purge data though modern implementations are often optimised for streaming data.

There's a debate about whether column databases are a type of NoSQL but given that you can use SQL and manage the database schema in the same way as relational databases it makes sense to consider them as a separate category.

Examples

- Google Bigtable: "a compressed, high performance, proprietary data storage system built on Google File System" (Wikipedia)
- Apache Cassandra: "a free and open-source, distributed, wide column store, NoSQL database management system designed to handle large amounts of data across many commodity servers, providing high availability with no single point of failure" (Wikipedia)
- Apache HBase: the Hadoop database (we'll look at this in the next topic), a distributed, scalable, big data store.

As you can see most of these are intended for high-performance, highly scalable applications.

References

https://database.guide/what-is-a-column-store-database/

Data Warehouses (DW) and Massively Parallel Processing (MPP)

The third class of storage options that we are looking at is data warehouses and the massively parallel processing options classically used in business solutions.

This topic is a specialist area in itself. We are only interested in understanding the general properties and uses of data warehouses and some of the terms used. There are a lot of acronyms!

A Data Warehouse (DW) is:

- a centralized repository
- stores data from multiple information sources
- data is transformed into a common, multidimensional data model
- highly efficient querying and analysis

This acts as or includes Massively Parallel Processing Databases (MPP):

- optimized to be processed in parallel
- many operations performed by many processing units at a time.

Data Warehouses are often used for business intelligence analytics particularly where there are large volumes of historical data.

Two terms you'll often see used with DW are OLAP and OLTP.

OLTP: **O**n-**L**ine **T**ransactional **P**rocessing, Operations: INSERT, UPDATE, DELETE
OLAP: **O**n-**L**ine **A**nalytical **P**rocessing, Information: complex analytics, aggregations, batch

Data warehouses offer OLAP and differ from transactional processing.

OLTP	**DW/OLAP**
● many single-row writes ● current data ● queries generated by user activity ● < 1s response times ● 1000's of users	● few large batch imports ● years of data ● queries generated by large reports ● queries can run for minutes/hours ● 10's of users

Generally, OLTP can support big data for many concurrent requests for small amounts of data each time while DW/OLAP can support big data for low concurrent requests for very large amounts of data each time.

You can learn more about data warehousing using some of the introductory descriptions given below. For this course you only need to have a basic understanding of the characteristics and when they might be an appropriate data storage choice.

References

- Datawarehouse4u.info
- Oracle.com

NoSQL

NoSQL is a very loose term for data storage methods that don't fit into a traditional relational database model.

It is sometimes referred to as "No SQL" to indicate that these DBMS don't use the traditional query/update methods or "Not Only SQL" to indicate that they offer additional query and optimisation functions.

NoSQL databases became popular with the increasing focus on big data and the need for efficient and powerful systems to store large volumes of the new unstructured or semi-structured data. They also allow for efficient, "scale-out" architectures instead of expensive monolithic architectures.

Generally, NoSQL databases have the following characteristics in common:

- Non-Relational
- (often) Open Source
- Schema-Less -- so adaptable to changes in the data structuring
- Horizontally Scalable -- to increase capacity you can add new separate servers rather than increasing the power of the existing server
- Lack of Adherence to ACID Principles -- ACID principles (atomicity, consistency, isolation, and durability) are the accepted properties of any transaction run on a DBMS
- No Standard Query Language -- databases often define their own specialist query language

Can be sub-classified into four types though many implementations offer a blend or mix of services:

1. Key-Value
2. Document store
3. Column store
4. Graph

Here we'll look briefly at the characteristics of key-value, document and graph data stores.

Key-Value

A key-value store is the simplest data model. Technically it is just a distributed persistent associative array (like a Python Dictionary). The key is a unique identifier for a value, which can be any data that an application needs to be stored. It's good for data sharing between applications and is a very fast way to get a value when you know the key.

Examples

- Redis: open source, in-memory data structure store
- Oracle NoSQL Database: JSON, Table and Key-Value datatypes (so also Document)
- Voldemorte: distributed key-value storage system, used by LinkedIn

Document

Document store is a data model for storing semi-structured document object data and metadata. The JSON format is normally used to represent such objects. You can query for documents by their content or metadata but you don't need to adhere to a strict scheme. That is, the content of the individual documents can be different.

Examples

- MongoDB: "general purpose, document-based, distributed database"
- Amazon DynamoDB: scalable database service by Amazon (also key-value)
- Microsoft Azure Cosmos DB: scalable document DB service by Azure (also key-value)
- CouchDB: Apache scalable native JSON-document store

If you'd like to try out MongoDB then you can use an online installation to navigate, load data and query a MongoDB installation from the command line in your browser.

Graph

Graph data structures store information as a set of nodes and edges. For example, social networks where each person can have a set of friends. A graph database is optimised to store and query graph structures. To get a view on the development and use of graph databases (from a Neo4j developer) read this article.

Examples

- Neo4j: a graph database focussing on data and its relationships at speed and supporting graph algorithms. Try it using the online sandbox at https://sandbox.neo4j.com/.
- dgraph: "a horizontally scalable and distributed GraphQL database with a graph backend".

Exercise

Can you find an example NoSQL system in the db-engines.com ranking list that's not mentioned here?

References

- What is NoSQL
- NoSQL Database types
- Benefits of NoSQL
- EMC Education Services, Chapter 10: Advanced Analytics—Technology and Tools: MapReduce and Hadoop, "Data Science and Big Data Analytics : Discovering, Analyzing, Visualizing and Presenting Data" DCU Library: https://capitadiscovery.co.uk/dcu/items/dda-17/EBC1908952

Big Data Storage

Remember that big data is defined as having high *volume*, *variety* and *velocity*.

Modern relational databases, data warehouses, columnar databases and NoSQL (Key-Value, Document) can manage very large volumes of data. The schema-less options of NoSQL can adapt well to the variety of big data, and the efficient implementations of key-value and columnar databases are adept at ingesting high-velocity, streaming data. However, there are other paradigms for processing, indexing, analysing and storing big data.

Where data is too large to be contained within a single computer system -- either the working memory (RAM) and/or the hard-drive storage -- then alternative methods for indexing and analysing the data need to be used. What can you do if the readily available computing power is small to moderate (due partly to cost of high-performance systems) and disk space (storage) is relatively cheap? How do you construct **algorithms** to make the most of available **resources** to handle big data?

Here we look at the Map/Reduce paradigm and then specifically at the Apache Hadoop Ecosystem and ELK as examples of tools for big data.

Map/Reduce

Let's look at the concepts behind the Map/Reduce paradigm and how to think about tackling a data storage and processing problem using this method.

Imagine you have a lot of data to process but you don't have a computer system. You have handwritten pages with the values you are interested in (perhaps a survey) that need to be turned into a statistical description. You have a team of mathematicians who can perform the required calculations and a floor where one room contains filing cabinets with the data sheets and each mathematician has a room where they can do their calculations. What algorithm do you use to get the required statistical description of your data?

Before Map/Reduce large-scale data processing was difficult and required:

- Managing hundreds or thousands of processors
- Managing the parallelisation and distribution of processing
- I/O scheduling challenges
- Implementation of status and monitoring tools
- Handling of fault and crash tolerance.

Map/Reduce is a programming paradigm, used, for example, by Google, for processing and generating large data sets with a parallel, distributed algorithm on a compute cluster. The data processing is broken up into two stages: Map and Reduce. There are many popular implementations on common cloud or hosting systems. The key contributions of Map/Reduce are:

- many small machines (often consumer grade hardware) tackle jobs not possible with larger more sophisticated systems
- scalability -- buy more cheap computers to increase capacity
- fault-tolerance due to built in redundancy and data management
- optimisation of execution

In Map/Reduce paradigms both computation and data storage are distributed.

Map: analyse and evaluate data, parse and filter or sort the input and produce key/value pairs.
Reduce: consume the key/value pairs from the map function and perform the required summary operation. This function starts with a very large number of key/value pairs and produces a much smaller aggregated set of key/value pairs.

Let's look at a simple example where we are trying to count the words used in a set of text files.

Map: reads in text and creates {<word>, #} pair for every word read
Reduce: takes all of those pairs and counts them up to produce the final count

The figure below is a graphical illustration of this process from the documentation of the Azure cloud platform.

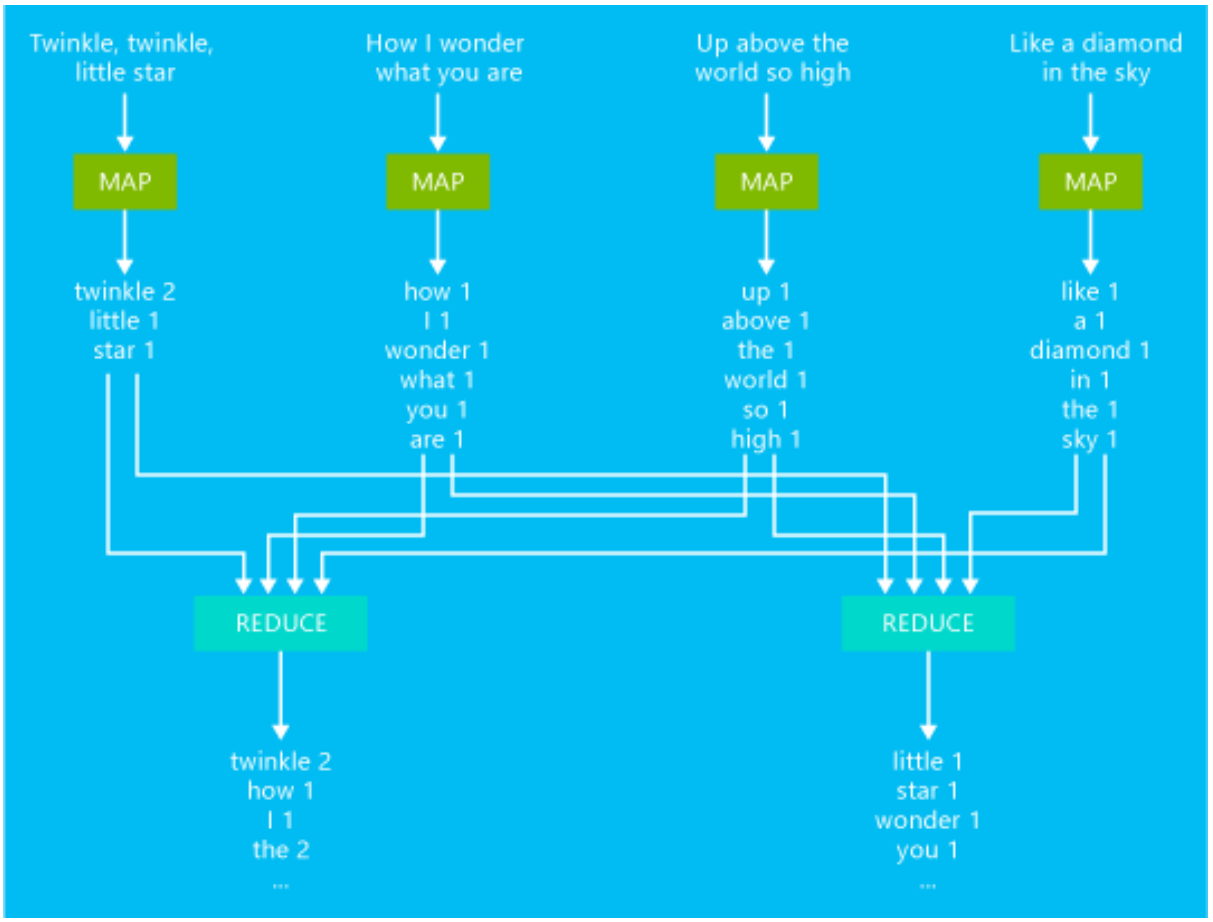


Fig: Map/Reduce analysis of "Twinkle, Twinkle Little Star" to produce a word count. (From: What is Apache Hadoop in Azure HDInsight?)

This tutorial gives a more detailed explanation of Map/Reduce and some example Java code.

If we go back to the example we started with, to manually analyse a large quantity of handwritten survey data, how could you go about this task in a Map/Reduce manner?

The Map/Reduce paradigm combines with a range of supporting technologies in the Apache Hadoop stack for big data.

Apache Hadoop

Now that you've got some idea about how Map/Reduce works, let's take a look at an implementation that uses it: the Apache Hadoop Ecosystem.

Apache Hadoop was first released in 2011 (v1) through to v3.3 in 2020. Apache Hadoop is "a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models". Hadoop provides both a computing (Map/Reduce) and a storage (HDFS) platform plus a collection of tools for Big Data processing, storage, management, analysis and more. The advantages of using Hadoop over traditional data warehouses which offer similar services include:

- cost optimisation,
- scalability,
- speed and
- flexibility.

A good introduction to the ecosystem can be found here. If you're familiar with SQL and RDBMS then this video (~6mins) gives an introduction to Hadoop by looking at the differences. For a more complete overview see Achari (2015) from the DCU library.

Hadoop operates as part of an ecosystem or a stack with other tools providing services such as data ingestion, coordination, scheduling, stream processing, search, data analytics, machine learning and more. The figure below shows a variation of the Hadoop ecosystem or you can see an alternative.

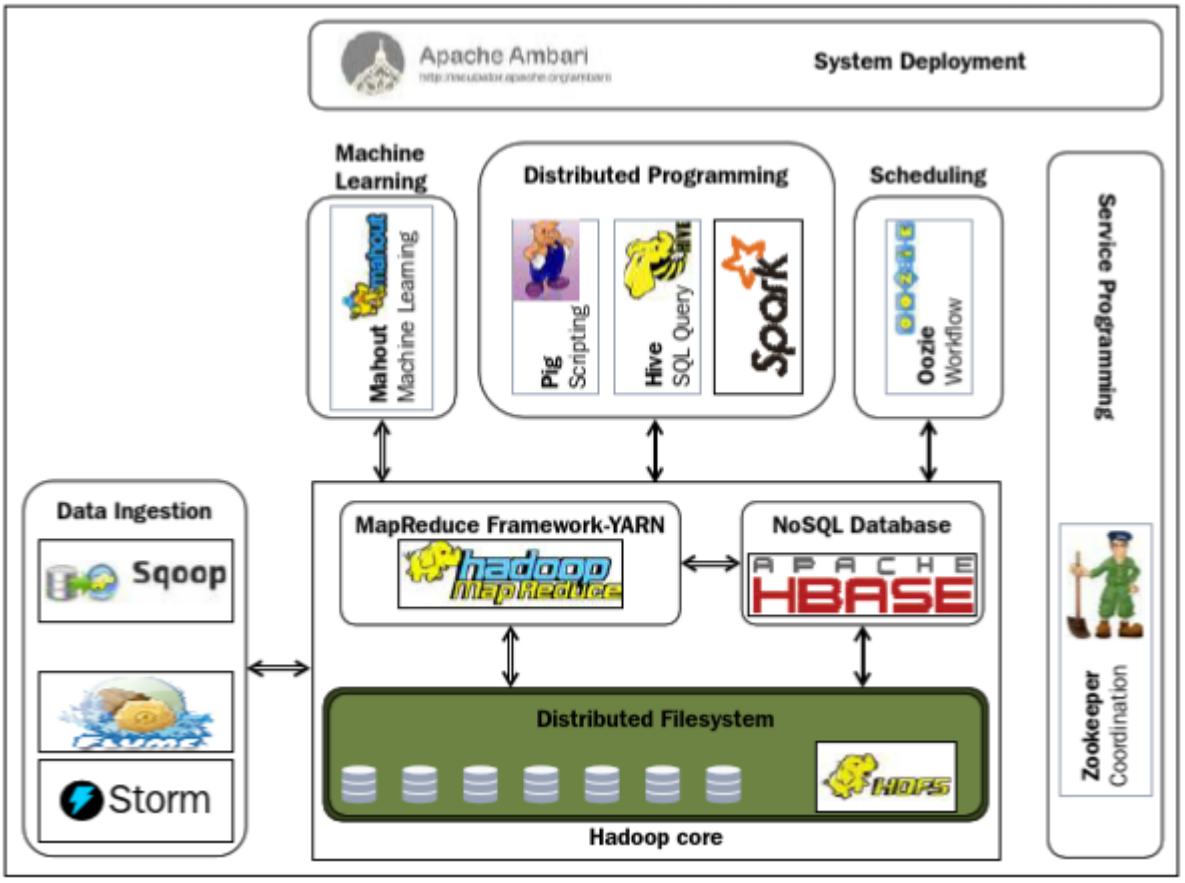


Fig: Hadoop Ecosystem (from: Achari, 2015)

Hadoop Distributed File System (HDFS)

Underpinning the Hadoop system is the Hadoop Distributed File System. This provides a number of key advantages, chief of which is the robustness. HDFS uses a master/slave model to mitigate against hardware failure and provide streaming access to storage for large datasets. It uses a simple model -- write once, read many -- and moves computation to the data rather than copying data to the processing. This enforces portability and heterogeneity. The diagram below shows the main components of HDFS.

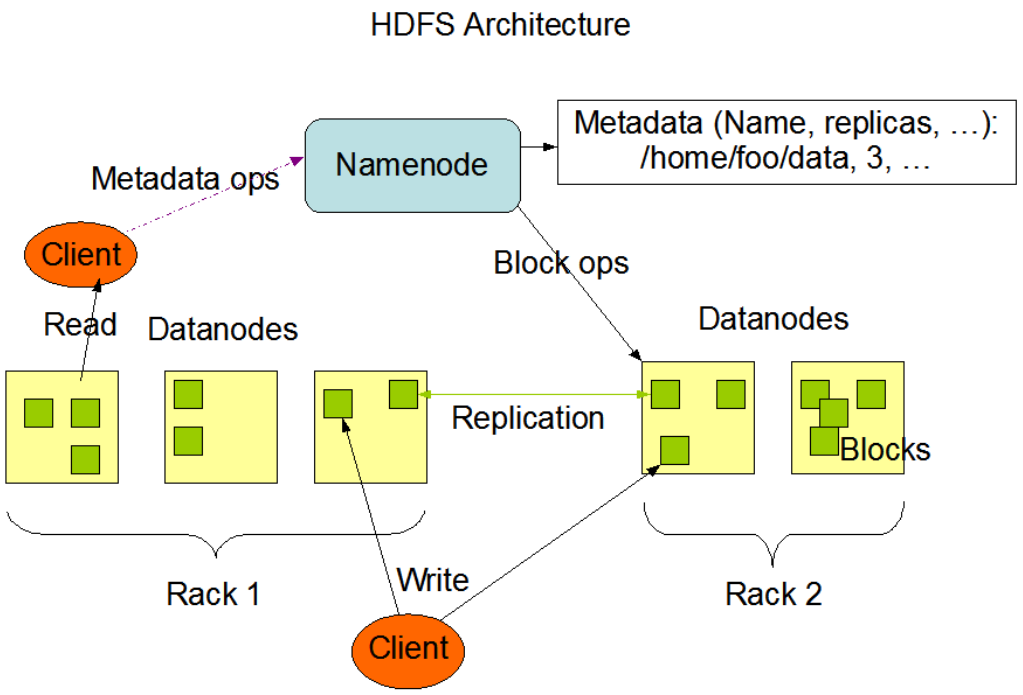


Fig: Main components of HDFS (from: Hadoop.apache.org)

Why is HDFS robust? The central idea is that large files are divided into **blocks** that are replicated across **DataNodes** and managed by **NameNodes**.

Data disk failure? The DataNode sends a continuous “heartbeat” message which is monitored by the NameNode that can then manage replication to replace the missing data.

Data corruption? For example, caused by network errors. The client computes and stores a checksum for each block when the block is first created. If the checksum fails then the correct data can be retrieved from an alternate DataNode containing a replica block.

NameNode failure? This used to be a situation that required manual intervention to recover from but since version 2 there is also replication of the NameNode to enable redundancy and recovery from failure.

Overall, HDFS is able to handle very large volumes of data with built-in options to manage most forms of failure or data corruption.

Exercise
It’s beyond the scope of this course to cover all of the tools and services in the Apache Hadoop stack. There are some simple exercises using Python notebooks if you would like to try Map/Reduce or using Hbase data storage.

References

Achari, S., 2015. *Hadoop Essentials*. Packt Publishing Ltd. Available as an ebook at [DCU Library](https://ebookcentral-proquest-com.dcu.idm.oclc.org/lib/dcu/detail.action?docID=2039889).

Elasticsearch & Elastic Stack (ELK)

Another common option you might find used for Big Data is a combination of elasticsearch and the elastic stack (formerly called ELK).

Elasticsearch is a distributed search and analytics engine that enables you to ingest JSON files and then perform full text search options across the documents. There are installations available in Amazon Web Services (and most other cloud services) and it's used with Apache Lucerne (indexing and search). Common use cases include log analytics, full-text search, security intelligence, business analytics and operational intelligence.

The ELK stack or Elastic stack is the combination of Elasticsearch, Logstash and Kibana plus Beats.

Logstash: server-side data processing pipeline that ingests data from multiple sources
Kibana: visualize data with charts and graphs
Beats: lightweight data processing (eg, 'tail log file')

In combination you could class this as a modern form of data warehousing for documents.

You can see from this document that most Big Data solutions are *combinations* of a number of complementary tools and services for data ingestion, processing and indexing to provide the required functionality.

Choosing a data storage approach

You might find yourself at the start of a data-driven project and need to make some recommendations for how to store the data that you will be gathering or generating. In this article, we look at some useful questions to ask yourself or your clients to determine your data storage requirements.

You've been introduced to the basic concepts behind a relational, column, data warehouses and NoSQL data storage options and you understand what makes "big" data. Here is a sample list of questions that you can use to specify your data requirements.

1. How much data do I have now? What rate will I get new data?
2. Is the data structured? What format is the data?
3. Do I have metadata or catalogue information? Is there a domain standard I can use?
4. Does the data need to be processed before loading?
5. How many queries will be run? Will they be concurrent? How many users?
6. What questions will the users be trying to answer? Do I know these questions?
7. Do I need to perform complex calculations or real-time analytics on the data?

Consider: Can you think of other important questions to ask at the start of a data-driven project? Do you work with databases? What do you think is the most important of these questions?

Decluttering

Data Visualisation has been following the Marie Kondo method since long before minimalism re-emerged as a design trend.

In the 1983 Visualisation classic “The Visual Display of Quantitative Information”, Edward Tufte introduced the idea of data-ink and the data-to-ink ratio where data-ink is the minimum ink that would remain on the page to maintain the same data message. In other words, what is left after all redundant and non-data ink has been removed. In digital media this can be referred to as the pixel-to-ink ratio. Tufte’s view is that you should aim for a very high data-to-ink or pixel-to-ink ratio. This is a highly minimalist approach to creating visualisations.

There are counter arguments for including non-data and decorative elements but most graphs can certainly do with some decluttering. Knaflic (2015) defines clutter as “visual elements that take up space but don’t increase understanding” and her book has a chapter on when and how to reduce clutter in your graphs.

Why should we apply a minimalist approach or think about reducing clutter? Remember how much is happening in our eyes and brains when we view, understand and interpret a visualisation. Any elements that don’t contribute to our understanding are adding unnecessary cognitive load on your audience. Even if they have an external motivation to understand your graphic (eg, so they can save money on their groceries or because they have to as part of their job), you should still aim for easy to view and interpret graphics.

Knaflic has some good examples of removing clutter from graphs. The figure below has been decluttered by following six steps:

1. Remove chart border: use whitespace effectively instead.
2. Remove gridlines: unless your audience needs to look up specific values from your graph
3. Remove unnecessary data markers: or at least use them with intent
4. Clean up axis labels: you don't need trailing zeros
5. Label data directly: why make your audience look up a legend when you can label the line
6. Leverage consistent colour: reinforce the relationship between line and label (look at Gestalt principles!).

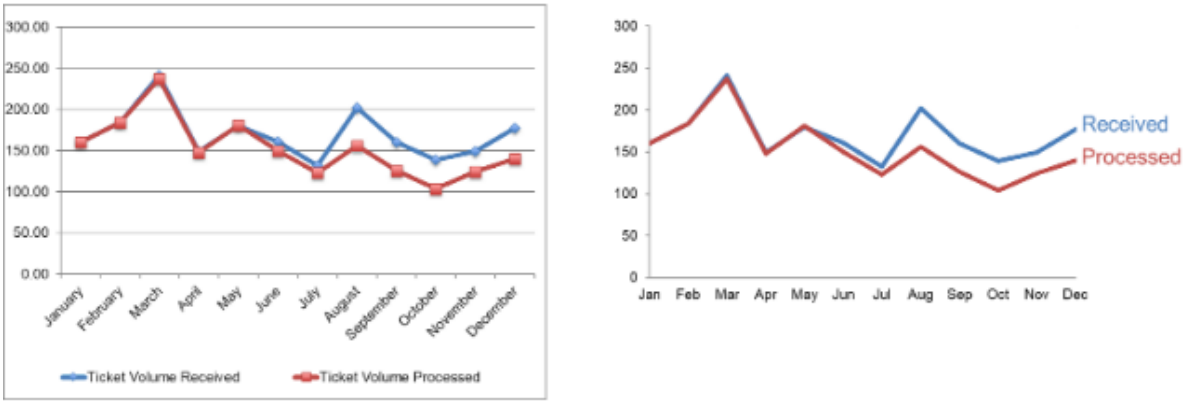


Fig: Before and After comparison of decluttering a line graph (Figure 3.24, Knaflic, 2015).

Another example made as an animated GIF is shown below. This comes from Darkhorseanalytics and there are also variations for tables, pies and maps.

Fig: Animation showing decluttering a bar chart (from: Spreakerdeck.com)

Decluttering is an easy way to improve your visualisations, removing distractions and creating a modern feel. Remember this point:

> *"The next time you are trying to improve a chart, consider what you can take away rather than what you can add."*
> (Data looks better naked — Darkhorse Analytics \| Edmonton, AB)

Discuss: Are you a minimalist or do you prefer some bling? There's an argument to be made for decorative or attractive non-data elements in data-driven visualisations for journalism or other mediums that need to engage the audience.

References

Knaflic, C.N., 2015. Storytelling with data. Hoboken: John Wiley & Sons.

Five Good Things to Know

If you don't remember anything at all from CA682 then try to remember these five simple tips for improving your data visualisations.

1. Avoid pie charts
2. Don't use 3D effects in a 2D medium
3. Be careful about Axes
4. Watch out for distorted Area
5. Distraction and Clutter -- "Remove to Improve"

If you've been following along the course then hopefully you'll recognise some of these and have some good ideas of the reasons behind them. Let's take a closer look at each of the tips.

1. Avoid pie charts

I've already mentioned that Pie charts are highly problematic. Recall: Edward Tufte says that pie charts "should never be used," (1973, p178), Stephen Few (2012, p94) "I don't use pie charts and I strongly recommend you abandon them as well … pie charts communicate data poorly" while Knaflic (2016, p61) says "Pie charts are evil" though there's a moderating view from the Storytelling with data community. There are dissenting opinions (for example).

Stephen Few's 2007 article "Save the pies for dessert" has a great breakdown, with examples, of the problems with pie charts and the reasons not to use them.

Pie charts are used to break a whole (100%) into components and work by encoding data values as angles. So to compare the relative size of different classes the viewer needs to judge the size of the angle. We are pretty good at this for major angles down to around 90° (or 4-5 slices) but as the angles get tinier or the difference between them is smaller then we are not very good at distinguishing the values. Take a look at the image below. See how the bar chart shows three distinct datasets but you would struggle to detect this from the pie charts.

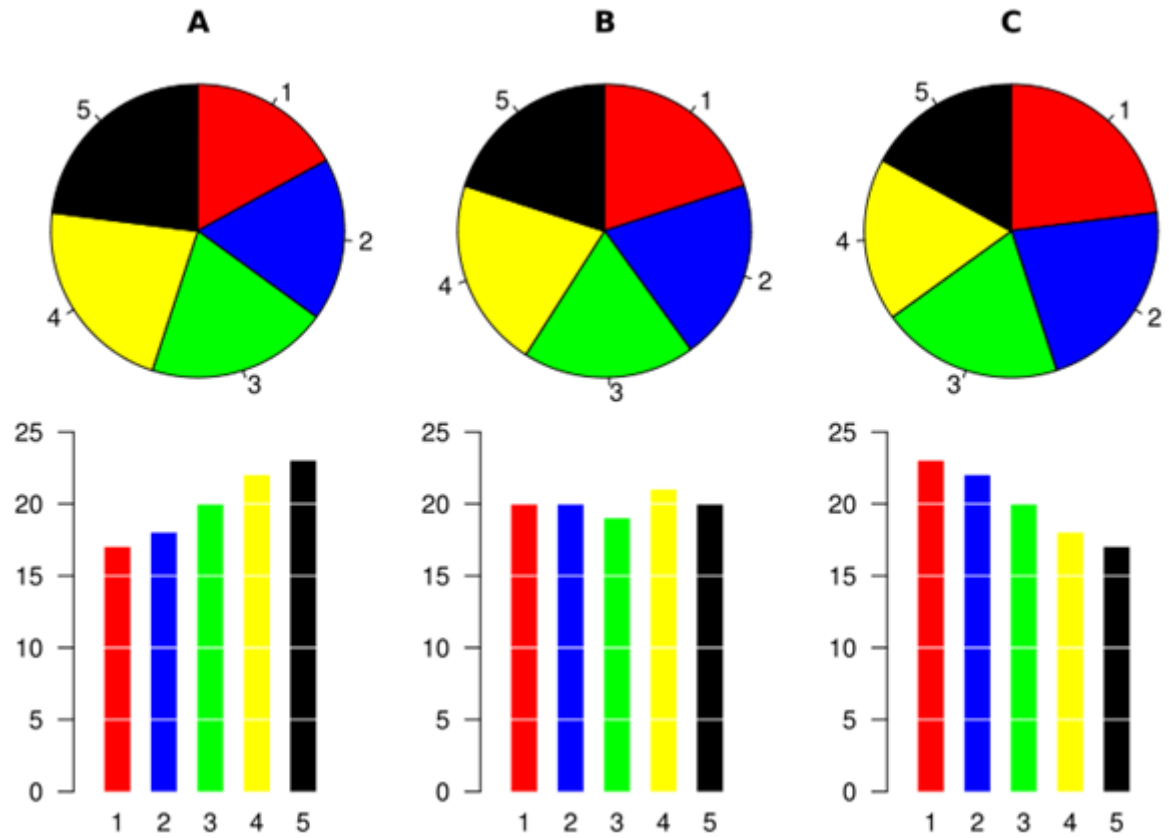


Fig: comparison of three datasets displayed using pie charts or bar charts (Source: wikipedia)

You often see pie charts used for data that doesn't add up to 100% or where there are dozens of different classes and many slices to try and read. Depending on how the chart is labelled they often over use colour and have complicated legends making it harder to see how the category label links to the pie slice. Can you find an example?

Pie charts also score poorly on the data-to-ink or data-to-pixels ratio and the same data can often be represented more effectively using other graph types. To avoid a pie chart check out these five alternatives.

A fairly balanced review of the strengths and weaknesses of pie charts using Tableau can be found here. They are occasionally either useful (where you have a small number of categories with very distinct differences) or unavoidable (your boss says you have to).

If you absolutely must use a pie chart then keep in mind the following principles:

- Only use for parts of a whole relationships (ie, 100% of something is divided up between categories)
- No more than 5 slices. If you have more than 5 categories then group or cluster them.
- Label carefully and order the slices increasing size clockwise as this minimizes the user effort.
- Never 3D and exploded isn't good either! Exploding the slices just makes the angles you're trying to compare further apart. But it looks cool :-(… just say no!

A variation of a pie chart is a doughnut or donut chart (example below) but this is generally worse for interpretability. The claimed advantage is that you can put a value or a statement in the empty space. However while the data-to-pixel ratio is improved, a doughnut chart removes the explicit indication of angle needed to easily compare the size of the sections making it even more difficult to read.

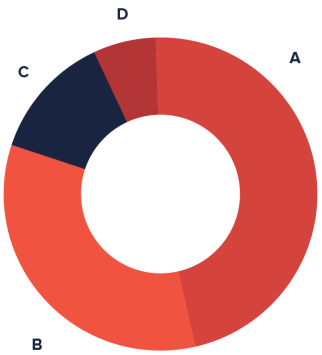


Fig: example doughnut or donut chart. A variation of a pie chart.

2. Don’t use 3D effects in a 2D medium

I’ve already said that using 3D effects in pie charts is not a good idea and if you think back to our discussions on how we perceive depth then you may recall why using 3D effects in general is a bad move. 3D effects in a 2D medium (like print or a computer screen) work by exploiting how we see depth. Artificially adding decorative 3D makes our brains work harder and using the virtual 3rd dimension for position can obscure data as shown in the example graph below. Can you work out the value for tom/east?

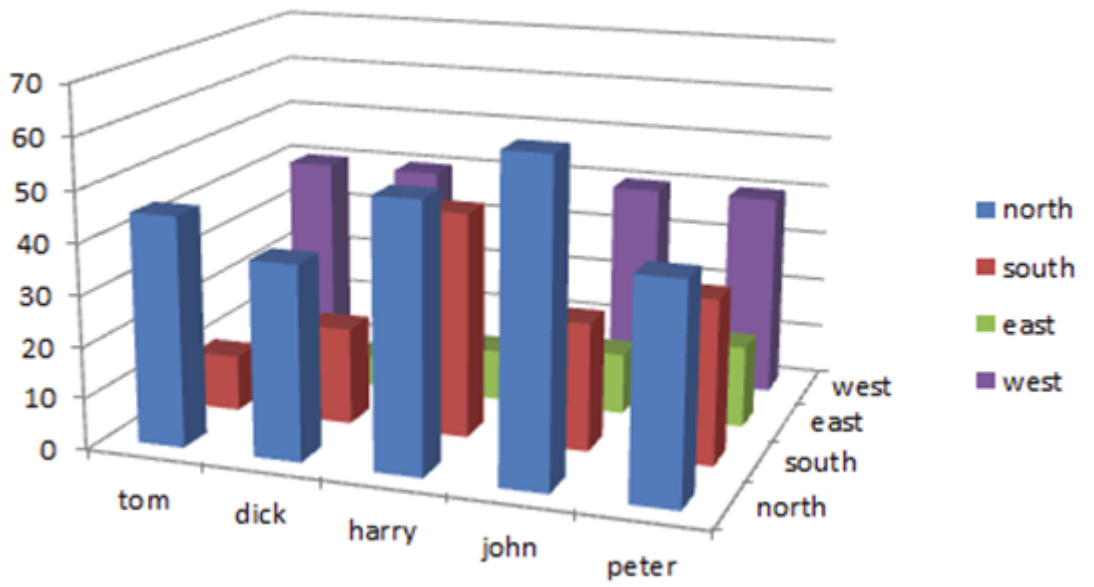


Figure: A 3D bar graph

However, if you are working in a Virtual Reality environment where the audience can interact with, move around and manipulate your graphic then using the third spatial dimension can be a valuable option for complex data. Likewise if you have interactivity on a computer screen and can pan, tilt, scroll then it can be useful. A visualisation that is mostly in print or 2D media? Don’t do it.

3. Be careful about axes

Almost all the graphs we look at have axes -- Y or vertical axis and X or horizontal axis.

A couple of general points of good practise with the axes are to always label them carefully and clearly, pay attention to the units and indicate them where necessary and be careful with exponential scales. Exponential or logarithmic scales are useful where you have a skewed distribution with a few very large values. However they can also be misleading if not properly marked. The viewer's expectation is that the divisions of the axis are regular and predictable and match to the tick spacing.

An important point to watch for is axes that do not start from zero. This is especially problematic in bar charts as shown in the diagram below. It can easily be used to exaggerate a change or if the baseline (x-axis) is blurred to reduce the appearance of the change. There are occasions where it makes sense to start the axis values from a non-zero value -- commonly seen in stock price line graphs where the interesting changes are at a high numeric value -- but be thoughtful in using it, especially in bar charts, and wary if you spot other graphs not starting an axis from zero.

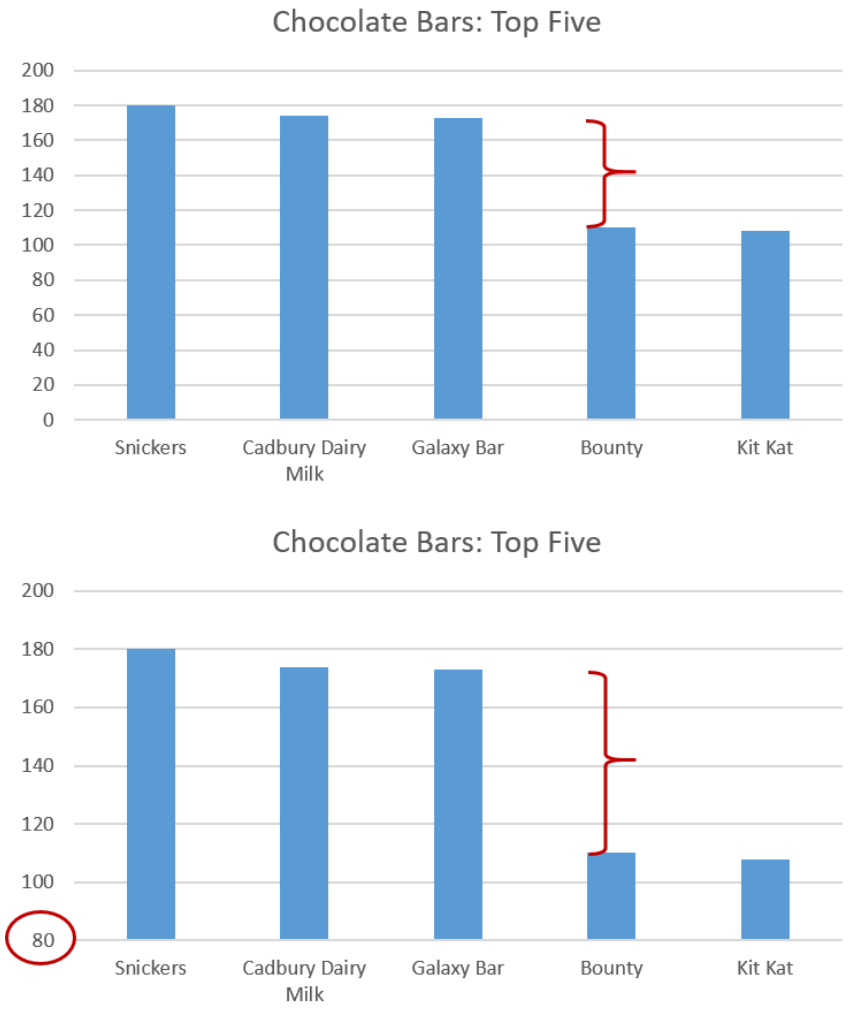


Fig: We can easily make Bounty & Kit Kat appear to be a lot less popular than the top three by changing the start point of the Y axis to exaggerate the height difference of the bars.

4. Watch out for distorted Area

This tip is less likely to be a mistake that you make but it is something to be aware of when looking at graphs.

When calculating the area for bubble charts or other graphs where your mark is circle with area make sure to use the area formula (πr^2) and not just set the radius or diameter of the circle to your data point. Most libraries or applications for visualisation will handle this properly for you but some do not. Why does this matter? Well look at the illustration below to see how the relative size of the circles can be misleading if you calculate the area based on diameter. In the first row of circles is the area of “5” really 5 times the size of “1”?

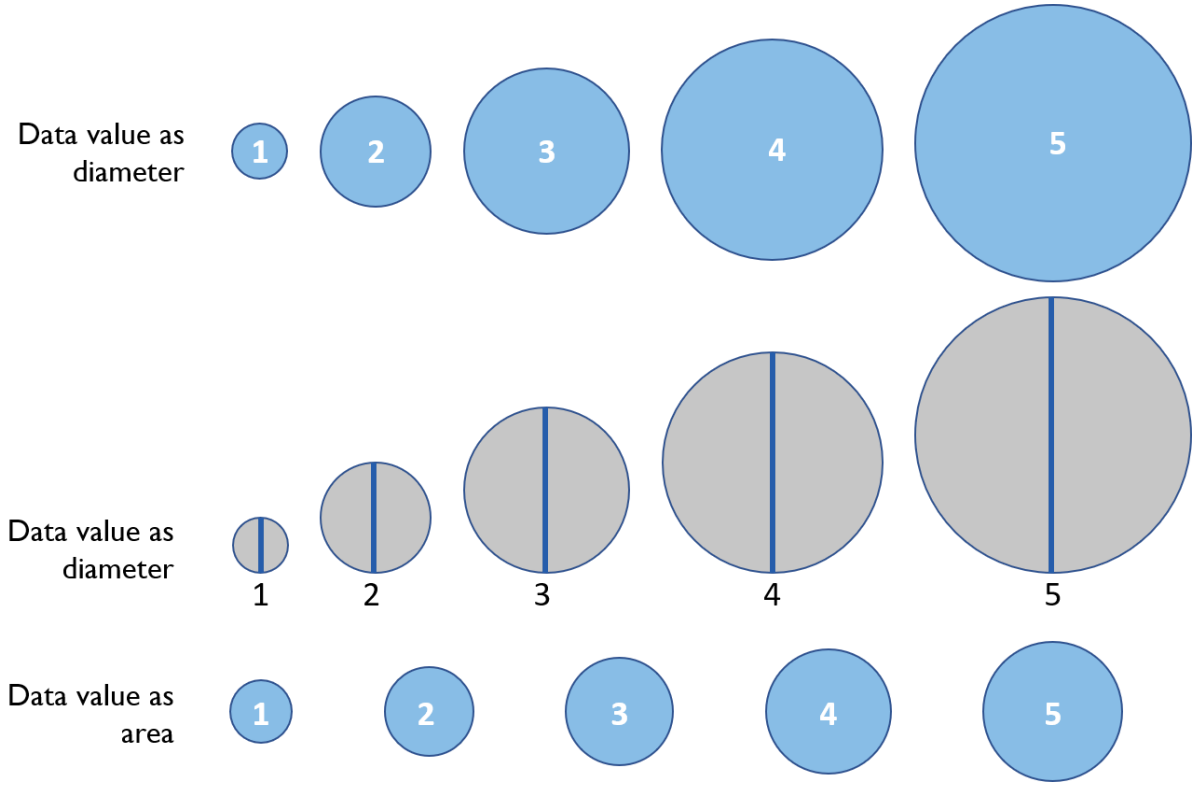


Fig: Comparison of the difference in circle appearance when using the data value as diameter compared with the correct method of data value is circle area.

5. Distraction and Clutter -- “Remove to Improve”

The final simple tip to remember is “remove to improve”. See the discussions on decluttering and attention.

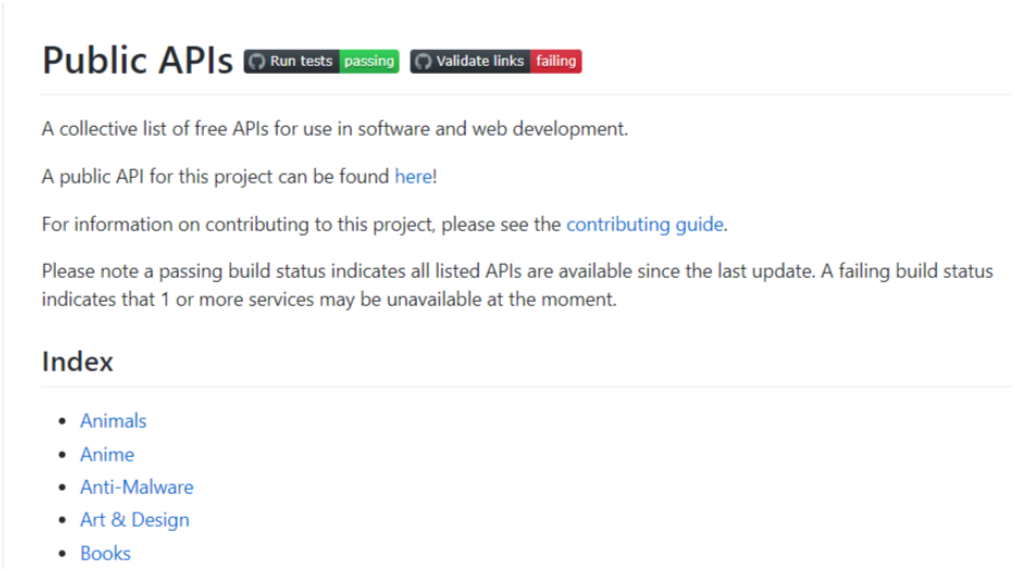
Final points

A bonus three tips to keep in mind as motivation for all your data-driven visualisation efforts.

1. Know why you're visualising – to understand or communicate?
2. Think of being correct & clear then be compelling (beautiful).
3. Be kind to your viewers.

Want more simple tips? See this infographic with 12 easy ways to improve your graphs.

Getting data from APIs



Public APIs

Run tests passing | Validate links failing

A collective list of free APIs for use in software and web development.

A public API for this project can be found here!

For information on contributing to this project, please see the contributing guide.

Please note a passing build status indicates all listed APIs are available since the last update. A failing build status indicates that 1 or more services may be unavailable at the moment.

Index

- Animals
- Anime
- Anti-Malware
- Art & Design
- Books

Not all data can be simply downloaded via a link to a file. Sometimes we need to query an online database or data storage application. This is an example of using an API (Application Programming Interface) to send commands to a programme and get a response, often a data fragment, in reply.

REST (REpresentational State Transfer) is a particular type of API that has become more common in recent years. You may have also heard of SOAP (Simple Object Access Protocol), which is a more powerful and consequently more complex standard for defining and implementing programming interfaces that can be accessed via web services.

Another common acronym that you'll come across is JSON (JavaScript Object Notation). This data format is commonly used when returning results from REST calls.

To get an overview of the main concepts behind APIs and REST check out this article. To get a simplified discussion on the concepts behind REST, check out this article.

Watch this video to give an introduction to REST APIs and some examples of how they can be used. Note that some of the example services are no longer available or may require an account.

REST APIs can be very useful ways to get the latest data from web sites or social media. The advantage of using APIs like this is that the call for data can be dynamic and embedded into a programme or data pipeline using common languages such as python, javascript or R.

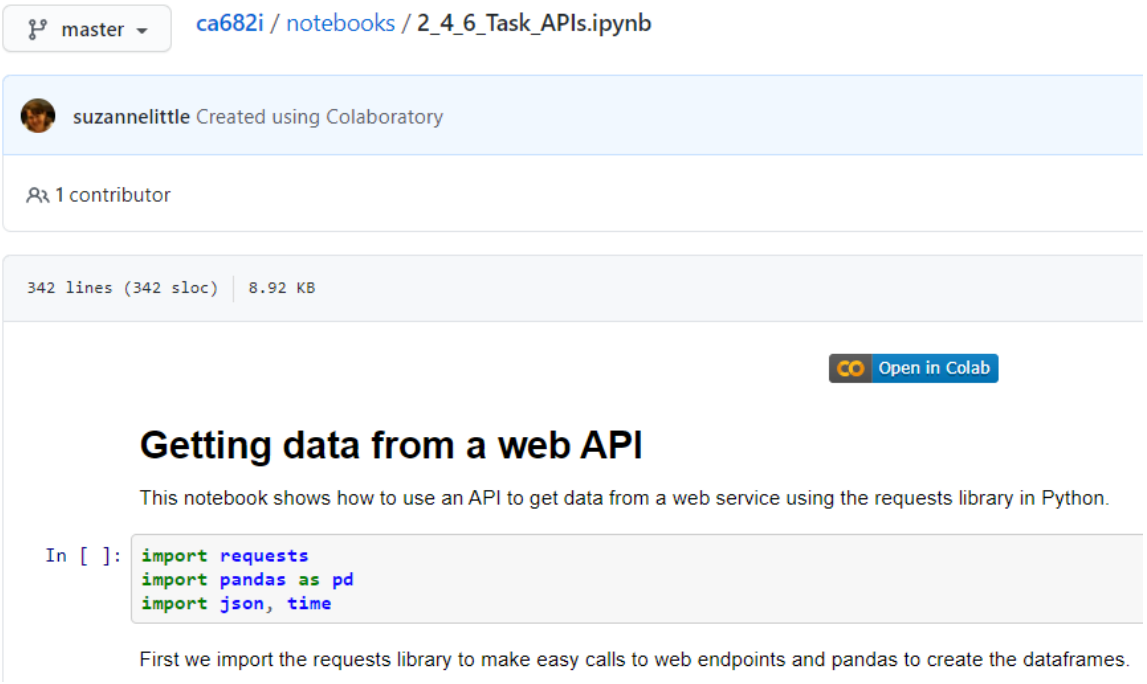
Consider: Can you think of some advantages and disadvantages of providing or accessing data via an online REST API?

Exercise

APIs for data access can be quite sophisticated. Here we will try a few simple options.

Firstly note that many APIs will require some kind of key or other authentication. This stops the server from being abused by many requests for data so check the terms and conditions when you request a key.

You can use programming libraries like curl or requests in python to make REST calls by creating the appropriate URL string. To fully explore or to develop REST apis an interface or browser plugin like postman, paw (Mac) or Advanced REST client can be used. Here we will use a Colab notebook.



The aim of this notebook is to try some simple calls to APIs using programming libraries.

Open this notebook in Colab and don't forget to save a copy of the notebook to your own account to work on it.

Further information on APIs and how they work can be found here.

Like to try some more APIs? Check out the Big List of Public APIs on github. Note that many of these will require registering for an access key and not all APIs are for accessing data.

How to use colour

We recognise values (colour, size, intensity) in comparison to the context. Think of a candle in a dark room vs one outside. The light emitting properties of the candle don't change (the physics) but our impression of its brightness certainly does. So how can we effectively use colour in creating visualisations?

We've already looked at:

- how humans see colour (RGB receptors),
- how colour is specified in computer systems (RGB pixels),
- how the hue, saturation and luminance are used as visual attributes to encode data and
- how colour is a preattentive feature.

Of course colour also has **emotional and cultural** meanings for the viewer so choosing appropriate colours for your visualisations should take into account many different properties.

First let's look at some of the key terms and concepts associated with defining colour.

Some terms

We've seen some of the terms relating to colour and when we looked at the use of colour for attributes to encode data. Just as we can define a colour based on its mix of Red, Green and Blue values (RGB colours), we can also specify any colour using Hue, Saturation and Luminance values (HSL colours). A HSL colour picker will let you see how each of these values changes the colour appearance. Hue, Saturation and Luminance are useful for understanding colour choices and the relationships between colours.

- **Hue**: different colours as measured using a colour wheel 0-359°. For example, 0° is Red, 120° is Green and 240° is Blue.
- **Saturation**: the vividness or "purity" of a colour, between 100% (pure colour) to 0% (grey). Also called Intensity.
- **Luminance**: visually perceived brightness or lightness, between 0% and 100% where white is 100%. Also called Lightness.

The figure below shows the Luminance value for a set of colours (Hue) with constant saturation and lightness levels converted to greyscale. This shows the difference in Luminance values and you can see how even colours with very distinct Hues can have similar Luminance. Notice that yellow (94%) is very close to white (100%) and that red, purple and mid-blue are all between 50%-60%.

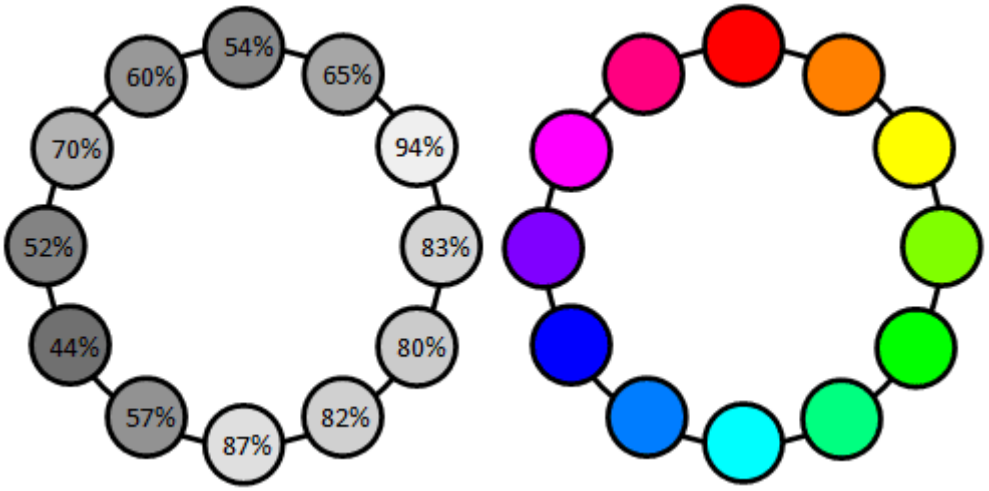


Fig: Luminance values of core colours (different hues) (from: http://www.workwithcolor.com/color-luminance-2233.htm)

Want to test your skills at matching colours according to Hue, Saturation etc.? Check out this online game - https://color.method.ac/. (I got 8.2!)

General tips on using colour

The main principle is to *use colour sparingly and thoughtfully*.

Remember that colour (Hue, Saturation and Luminance) is a data encoding attribute. Any use of colour on your graph can represent values from your data. For example, if you use different Hues where there is only one category then you are implicitly indicating that multiple categories are present and this is misrepresenting your data.

Colour is also strongly associated with attention and Hue and Saturation (Intensity) are both preattentive attributes.

So if you overuse colour you are diluting your message and overwhelming the attention channels of your audience.

Don't use multiple colours for the sake of being colourful. For instance, a bar chart with only one field on the categorical axis generally doesn't need to have separate colours for each bar.

The figure below shows an example (from Knaflic, 2015) where changing from a colourful graph (multiple hues, high saturation) to a monochromatic graph (one hue, varied saturation) improves the interpretability. Notice how much faster you can pick out the top performing sales regions (rank 1) from the graph on the right.

Country Level Sales Rank Top 5 Drugs

Rainbow distribution in color indicates sales rank in given country from #1 (red) to #10 or higher (dark purple)

Country	A	B	C	D	E
AUS	1	2	3	6	7
BRA	1	3	4	5	6
CAN	2	3	6	12	8
CHI	1	2	8	4	7
FRA	3	2	4	8	10
GER	3	1	6	5	4
IND	4	1	8	10	5
ITA	2	4	10	9	8
MEX	1	5	4	6	3
RUS	4	3	7	9	12
SPA	2	3	4	5	11
TUR	7	2	3	4	8
UK	1	2	3	6	7
US	1	2	4	3	5

Top 5 drugs: country-level sales rank

RANK	1	2	3	4	5+

COUNTRY | DRUG	A	B	C	D	E
Australia	1	2	3	6	7
Brazil	1	3	4	5	6
Canada	2	3	6	12	8
China	1	2	8	4	7
France	3	2	4	8	10
Germany	3	1	6	5	4
India	4	1	8	10	5
Italy	2	4	10	9	8
Mexico	1	5	4	6	3
Russia	4	3	7	9	12
Spain	2	3	4	5	11
Turkey	7	2	3	4	8
United Kingdom	1	2	3	6	7
United States	1	2	4	3	5

Fig: Example of how restraint in use of colour improves clarity (from Knaflic, 2015)

A takeaway message from this example is to **be careful using hue**. It is much harder to distinguish between differences in hue, especially when the object area is small.

Reserve the use of bright (highly saturated) colours for items that need attention or those that should be examined first, and use less intense (less saturated) colours for other items on the page. Overuse of hue and saturation will be distracting.

Few (2004) condenses these principles down to use cases for three palettes -- Light, Medium and Dark & Bright. The figure below shows example palettes used in “Show me the numbers” where the Hue and Saturation are kept relatively constant and the decreasing Luminance value creates the difference. Note that you would rarely if ever use all 9 colour options from the palette in a single visualisation.

- **Light:** use for large data encoding areas (bars, areas)
- **Medium**: use for small data encoding areas (points, lines)
- **Dark & Bright**: use as an accent to draw attention

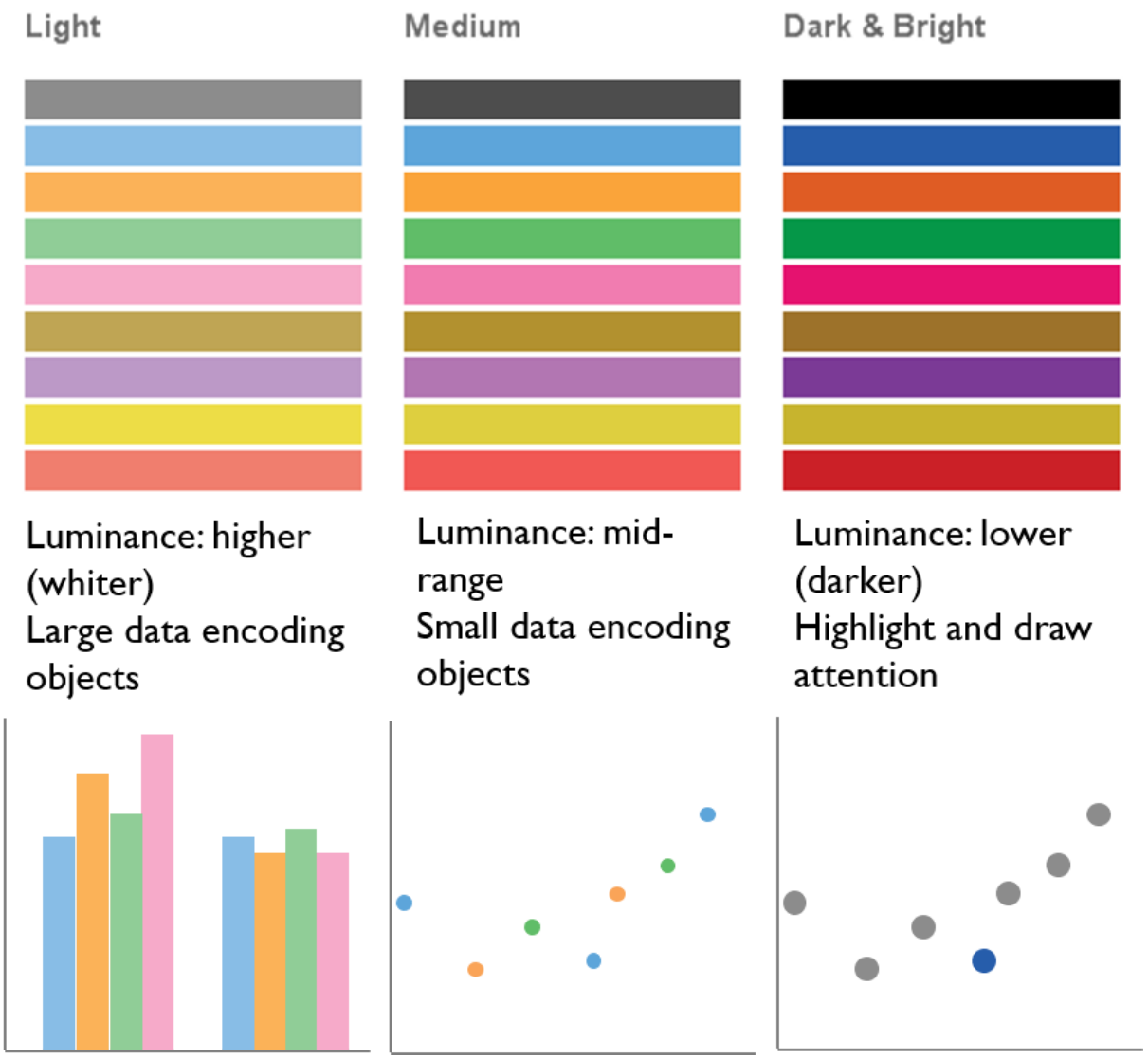


Fig: Three example colour palettes (Few, 2004) showing how variations in Luminance value should be used when encoding data in graphs.

An excellent, succinct and practical reference for using colour in data-driven visualisations is provided by Stephen Few - http://www.perceptualedge.com/articles/visual_business_intelligence/rules_for_using_color.pdf.

What about branding?

Maybe you're doing some work for a company or organisation that has a strong branding or logo that you need to reflect in the work. Knaflic has a good section on how to handle brand colours (Knaflic, 2015, p123). A simplified option is to try and use one brand colour as an action or highlight colour and, if necessary, reduce the saturation or increase the lightness to maintain the tie to the brand without over use of colour. You should still avoid unnecessary or misleading use of colour as a data attribute -- eg, different coloured bars where there is no difference in category.

Cultural influence on colour

Doctor Who “The Empty Child” (BBC, 2005):

Rose: What’s the emergency?
The Doctor: It’s mauve.
Rose: Mauve?
The Doctor: Universally recognized color for danger.
Rose: What happened to red?
The Doctor: That’s just humans. By everyone else’s standards, red’s camp. Oh, the misunderstandings. All those red alerts, all that dancing.

Colours have meaning quite separate from their physical properties. These can be seen in the names we assign to colours and can vary according to context and cultural background. There’s no universal meaning for colours but there are some strong cultural traditions and standard practise that you should be aware of. To manage these different interpretations of colour meaning you need to be aware of your audience and intended message and be thoughtful about the tone that colour conveys (emotion and cultural connotations). This is especially important in branding and logos.

You’ve already seen the XKCD color survey. Muyueh Lee has also done an analysis of the differences in English and Chinese colour names taken from wikipedia.

David McCandless has made a visualisation showing “Cultures and Color”. Can you find a colour that has a very different meaning between two cultures?

Choosing colours

If you are using data visualisation tools like Tableau then many of the default options will normally take into account good visualisation principles and colour contrast to ensure attractive and effective graphs. However, you shouldn't rely on the defaults -- especially in your visualisation assignment! -- but think carefully about what the colour choices are communicating. For your assignment make sure you justify the colours using some of the principles discussed in this article.

Summary of general principles:

- Use colour sparingly and intentionally
- Vary Luminance to highlight details
- Vary Hue to indicate different categories
- Low saturation for large areas of colour (eg, backgrounds)
- High saturation for small areas of colour (eg, marks, labels)
- Be aware of potential colour blindness issues and how your graph may be viewed in black and white
- Be aware of the difference in contrast between foreground (graph marks or labels) and background colours. Large differences in contrast values are easier to read. The figure below shows a high contrast (Left) and low contrast (Right) example.

Fig: Example of high contrast vs low contrast colours used for text and background.

Can you identify another problem with the choice of colours for the example on the right?

Resources

There are many resources on choosing colour -- for art and design, for logos, for web pages and for graphics. Here are some potentially useful websites and tools.

- API for checking contrast ratio - https://webaim.org/resources/contrastchecker/
- HSL Color Picker; HTML Color Picker; Web, HEX, CSS, HSLa
- https://contrastrebellion.com/ - a website outlining the problems with low contrast font colour
- http://paletton.com/ - for exploring and creating colour palettes (general design)

- https://colorbrewer2.org - for creating colour sequences for cartography (map) data. Seaborn (Python Data Visualisation Library) also has defined colorbrewer palettes.
- https://jiffyclub.github.io/palettable/ - “Palettable (formerly brewer2mpl) is a library of color palettes for Python.”
- https://www.checkmycolours.com/ - options for testing for colour blindness issues
- https://coolors.co/ - explore colour palettes and compare different colour specifications

Reflect

1. What’s your favourite colour? Give both the name you would use and the HSL values. You can use a colour picker to find the exact match.
2. Find a colour that has a different meaning depending on the context or culture - https://www.informationisbeautiful.net/visualizations/colours-in-cultures/.

References

Cole Nussbaumer Knaflic (2015) “Storytelling with Data” Wiley
Stephen Few (2004) “Show Me the Numbers” Analytics Press
Stephen Few (2008) Practical Rules for Using Color in Charts
Design Concepts For Better Power BI Reports – Part 2: Preattentive Attributes
Color Luminance: perceived brightness, luminance and contrast

Introduction to scraping data from websites

Recall that when I talked about possible sources of data one of the options was to scrape data from websites. Here we look at some of the pros and cons of doing this, general rules that you should bear in mind and some general tips on implementing a scraping script.

What is scraping?

Websites don't always provide their data for easy download or via an API as CSV or JSON. Or even as neat and tidy tables where you can try the pandas function read_clipboard() to copy and paste tabular data!

Sometimes you need to get dynamic data from a website but there's no API. If you know a little bit about how HTML works then you can use *scraping* to gather the raw data. Downloading and processing the HTML files is how search engines index the Internet.

> **Scrape:** to extract data from semi-structured sources (e.g., webpages).
>
> **Crawling:** traversing the web via links in <a> tags to gather data via scraping.

The general process is as follows:

1. Have a plan (how to identify the data items on the page)
2. Request webpage (e.g., urlopen, requests)
3. Parse HTML (e.g., lxml, beautifulsoup)
4. Store data (e.g., as list or dict)
5. Format as required (e.g., CSV, json, dataframe, sql)

You will almost certainly need to clean the data as scraping can be very prone to introducing errors and artefacts.

Good practise for scraping

Some good rules to remember about scraping (from Gregreda.com):

1. You should check a site's terms and conditions before you scrape them. It's their data and they likely have some rules to govern it.
2. Be nice! A computer will send web requests much quicker than a user can. Make sure you space out your requests a bit so that you don't hammer the site's server and cause a denial of service attack.
3. Scrapers break: sites change their layout all the time. If that happens, be prepared to rewrite your code.
4. Web pages are inconsistent: there's sometimes some manual clean up that has to happen even after you've gotten your data.

Python provides some handy libraries to help with scraping including:

- requests - downloading the page
- BeautifulSoup - parsing the HTML into an object to search and manipulate

These libraries are fine for once off tasks or exploring scraping but for more stable, longer term projects check out Scrapy. There are other tools for web scraping given in this ranked list.

Consider: Have you ever needed to get data from a website? How did you do it and did anything go wrong? If you haven't then can you think of when a web scraper might be useful?

Exercise

Here we try some simple web scraping to get a list of book titles and prices from an online store - http://books.toscrape.com/. This is a fake book store created to provide a demo web site to practise scraping.

View the Colab notebook hosted on Github to see the code. Can you alter it to include the star rating for each book?

> Open the notebook in Colab and don't forget to save the notebook to your own Google Colab account to save your work.

A challenge for you: Can you get a list of all the All Ireland Senior Football Champions (just the county name) from https://en.wikipedia.org/wiki/List_of_All-Ireland_Senior_Football_Championship_finals? Remember: don't hammer wikipedia by repeatedly using requests to get the page! Get it once and then work out your scraping code.

Resources

Some extra tutorials and guides if you are interested in learning more.

- Beautiful Soup: Build a Web Scraper With Python – Real Python
- Web Scraping 101 with Python

Metadata

Suzanne Little, suzanne.little@dcu.ie

Metadata can be simply defined as **“data about data”**. It’s an important part of managing data in both the Data Gathering & Data Preserving phases in our generic pipeline. But what does metadata look like, how is it used and what are some of the challenges in creating useful metadata?

What is it and why is it useful?

Let’s take a look at what we mean by “metadata” and why it might be useful.

If you ask a librarian, they might tell you that metadata is an “inferior type of cataloging”! However, in computing and data fields, metadata is “information about an object, be it physical or digital” that summarises the main qualities or attributes of the object. More formally, Baeza-Yates (1999) in “Modern Information Retrieval” defines metadata as “information on the organization of the data, data domains, and the relationship between them”.

You’ve probably come across metadata before in schemas, data descriptions, specifications or catalogue data.

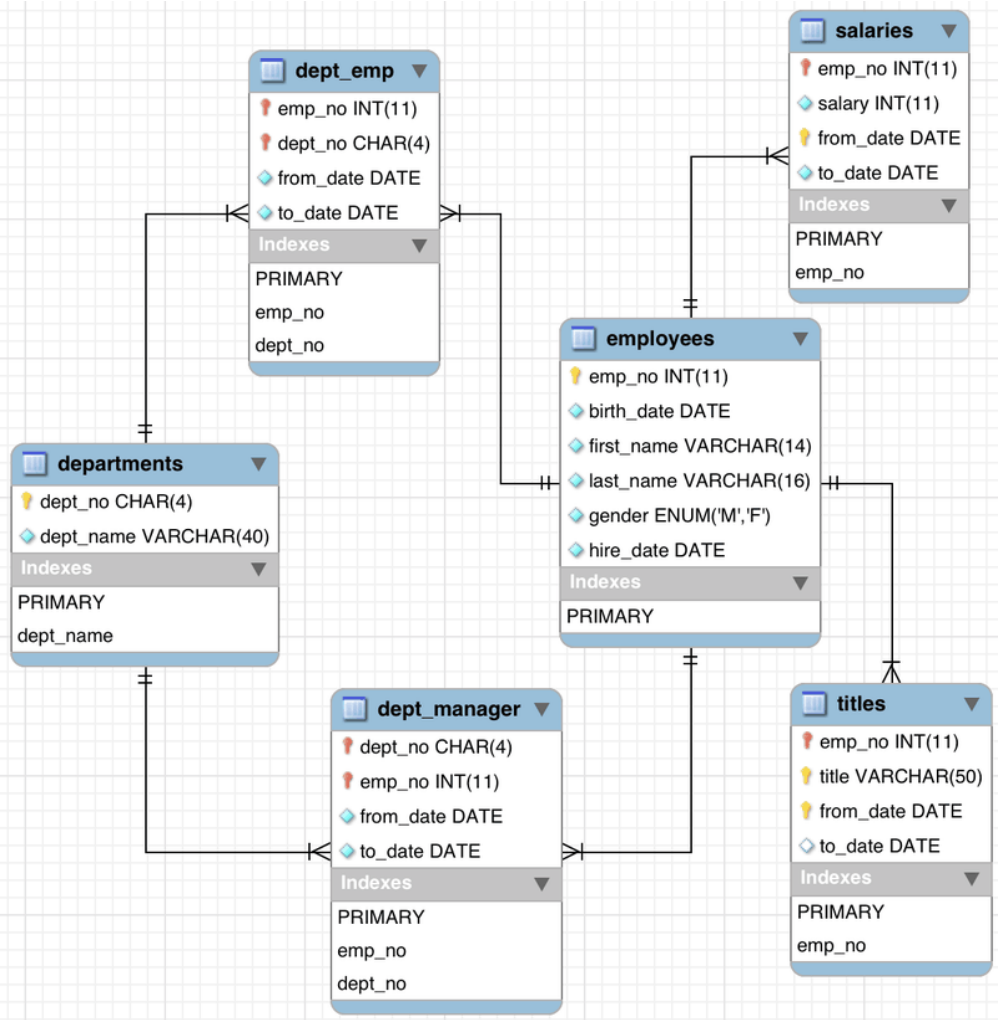


Figure 1: Database schema

Metadata is useful to help us to use data. Namely, metadata is used to:

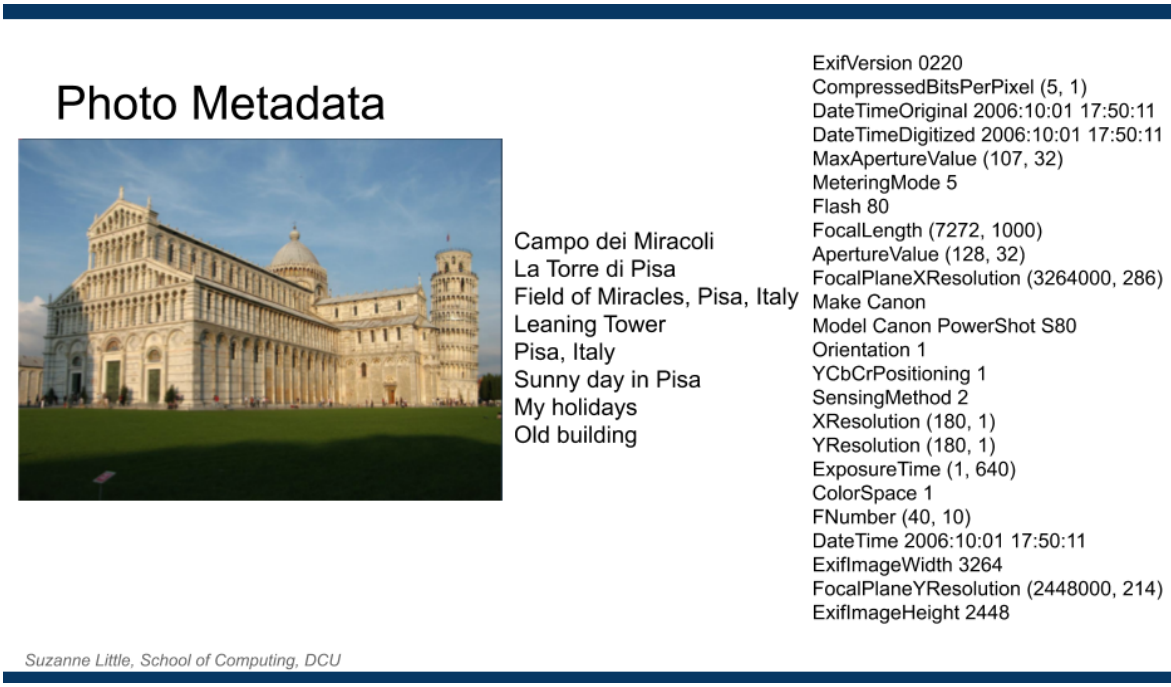
- Find
- Locate
- Identify
- Select
- Obtain
- Navigate

Importantly, metadata is also the place where information on rights management and privacy are often included.

There are different ways to think about metadata and explicit standards that you can use but it can be helpful to consider three different types of metadata:

Descriptive: what the information object is about; inherently intrinsic properties
Administrative: who, what, why, where of the object's creation and management; inherently extrinsic properties
Structural: information about the structure, format, and composition of the thing being described; can be intrinsic or extrinsic

Let's take a look at examples of metadata for a complex digital object. Watch the YouTube video at https://youtu.be/Ts-eUudbyLo which provides an explanation of photo metadata.



If you'd like to try extracting some EXIF from your own photos, there are various websites online that will do this. Or you can view the metadata fields using a photo editing tool like Adobe or GIMP or by right-clicking on the image icon in Windows Explorer, choosing *Properties* and then *Details*.

Exercise: Take a photo using your mobile or find one online and view the associated metadata using either an online EXIF tool or viewing the file properties. Is there any information that surprises you or seems particularly useful? Can you identify descriptive, administrative or structural metadata?

Keep in mind the difference between *data* and *metadata*. This is often an arbitrary distinction (When does metadata become data itself?). For this course, remember that metadata is the *names of the attributes* (author, date, title) and not just the values ("Suzanne Little").

Exercise: create your own metadata

Now that we know what metadata is and why it is useful. Try to create some metadata of your own. Follow the steps below and see what different aspects you think are important for using metadata.

1. Describe your favourite book or movie.
2. What qualities or attributes did you use? Title, Author, Year, Genre, Characters, ... ? These qualities define the metadata.
3. How would you use your “metadata” to Find, Locate, Identify, Select, Obtain and/or Navigate a collection of books or movies?

I’ll refer back to this exercise later in this document so keep notes on your metadata example.

Metadata Granularity

Metadata can apply to different dimensions or concepts:

Abstraction: how close the metadata is to the data or object.
Granularity: how detailed the metadata is.

Let's discuss **abstraction** first.

Imagine you have written a novel. It's wildly successful and a second edition is published plus an audiobook read by your favourite actor! Your story is the *work*. It is your artistic creation without any specific manifestation. The published story (the book) is an *expression*. This is the medium (channel) that expresses your work. The different published versions of the book are *manifestations* of this expression and the specific instance of the book (the autographed copy you keep on your shelf) is an *item*. The audiobook is a different example of an expression. In this case, there is one work with two expressions, one expression has two manifestations and there are many thousands of items. The table below describes this abstraction hierarchy using this example.

WORK ⇓ ⇓ ⇓	an abstract entity; the distinct intellectual or artistic creation; it has no single material manifestation	"The Very Clever Scientist" by Suzanne Little	Title="The Very Clever Scientist"; Author="Suzanne Little"
EXPRESSION ⇓ ⇓	the multiple realisations of a work in some particular medium or notation, where it can actually be perceived		Format="hardcover" or Format="audio/mp3"
MANIFESTATION ⇓ ⇓ ⇓	each of the formats of an expression that have the same appearance; but not necessarily the same implementation		Edition=1st; ISBN=978... or Edition=2nd; ISBN=978...
ITEM	a single exemplar of a manifestation; if we distinguish this level it is because otherwise identical manifestations have some differentiation		Owner="Suzanne Little"; Protective_Cover="plastic"; Autograph="Meryl Streep"

Why is it useful to think about the levels of abstraction of metadata? Well, particularly in examples like the one given here, you need to consider which level in the hierarchy you are describing when you assess the value and usability of the metadata. This is also true of other examples where you have a manifestation of a creative work or a digital copy of a created physical object. Does the description apply to the specific physical copy of the book or to all instances of the story? **Consider**: who is the "artist" if I take a photograph of a sculpture?

Abstraction is one way that metadata can be structured. When dealing with complex objects, especially digital representations of physical objects, you can see where there could be

different ways of describing the data. Another way of thinking of this is the **granularity** of the metadata descriptions.

Figure 1 below illustrates a granularity hierarchy that moves from very low-level *features* describing the digital representation (eg., things that can be calculated from the digital image pixels) to slightly more complex *structure* feature descriptions that are calculated from these values. The next layer describes the meaning (*semantics*) that are represented in the digital object. The "semantic gap" refers to the difficulty in automatically (i.e., without significant human effort) creating the metadata that describes the meaning of a digital object. Finally, there is information that generally needs to be recorded by a human, *bibliographic* or management data about the creation and instance of the digital object.

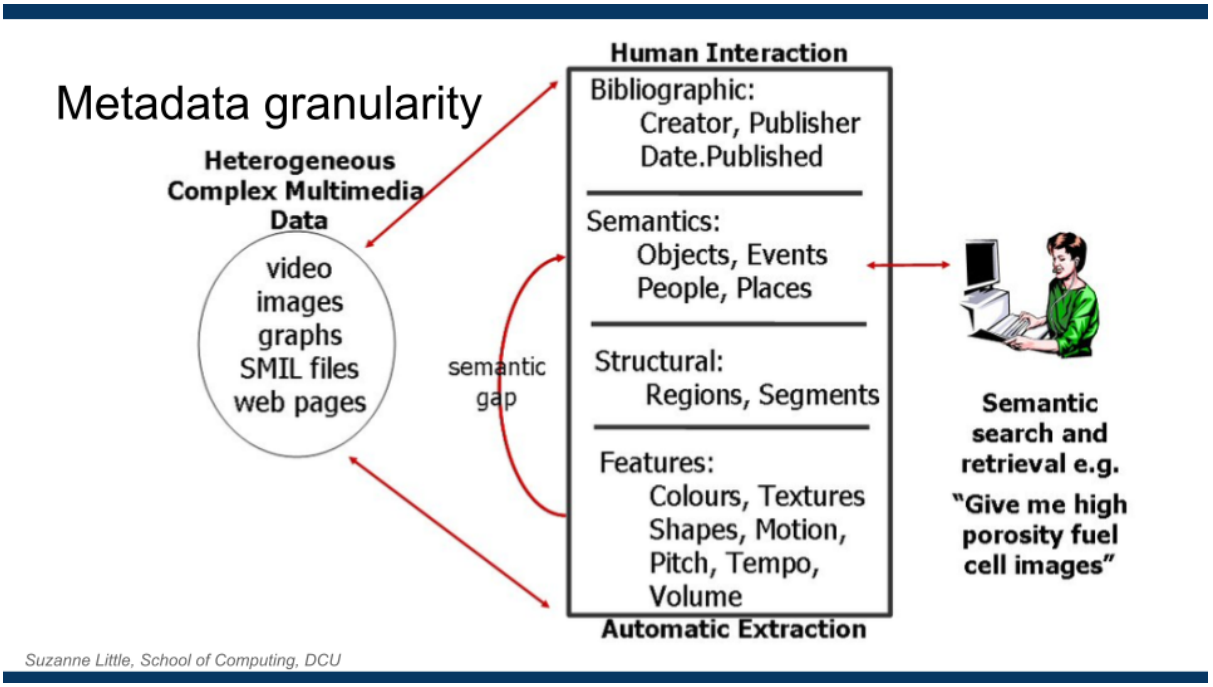


Diagram showing metadata granularity

Metadata Standards

"The wonderful thing about standards is that there are so many to choose from."

[This quote is most commonly attributed to Andrew S. Tanenbaum (1981) but also to Grace Hopper in The UNIX-HATERS Handbook (1994).]

You should now understand why metadata is useful (reminder: to Find, Locate, Identify, Select, Obtain and/or Navigate your data and for rights management) and you've made some metadata of your own at different levels of granularity and hierarchy.

If you haven't done the exercise yet then go back and do it now. Take a look at the metadata you created. What terms did you use? Would everyone have described their book or movie in the same way?

This is where metadata standards are helpful. Standards can cause some problems -- perhaps it doesn't include the information that you need or it's just too big and complicated for your application. This is where the quote above is often used in many different subject areas!

A metadata standard provides the structure, syntax, vocabulary and/or format to specify your data. This enables a common understanding of the meaning and allows the attributes describing the data (the metadata) to be used and interpreted.

Discuss: Can you suggest a problem if there are too many standards to choose from? What if every person in this class specified their own "metadata standard" for their student record?

Metadata standards can be simple and specify basic administrative metadata (see 1.4.2) and the *syntax* for the metadata or they can be more comprehensive and include elements such as controlled vocabularies or ontologies that define the *semantics*. That is the meaning and relationships of the allowed values for the metadata. For example, a syntax specification may specify that a mandatory attribute of the data is "Owner" with a value of type string but the semantic ontology could define "Owner" as a type of "Legal Entity".

There are also numerous bodies that create and publish standards including ISO, RFC, IEEE and W3C. Metadata standards might be produced by organisations like the Research Data Alliance, the Digital Curation Centre or government bodies. There are often domain specific organisations that create metadata standards for very specific applications like geosciences, biomedical, cultural heritage and economics.

To explore the many standards that are used, you can look at https://www.dcc.ac.uk/guidance/standards/metadata/list or https://en.wikipedia.org/wiki/Metadata_standard#Available_metadata_standards.

Here I'll discuss one of the most common, general standards: Dublin Core.

Dublin Core

Named after the location of the first workshop in Dublin (Ohio not Dublin, Ireland!) Dublin Core is a small set of vocabulary terms (DC) that can be used to describe both digital and physical resources. The semantics of DC was established by an international, cross-disciplinary group of professionals from librarianship, computer science, text encoding, the museum community, and other related fields and is endorsed as a standard by official bodies including IETF, ISO and NISO. It was proposed in 1995 as a standard set of metadata elements, simple enough to be supplied by the document's author rather than a professional curator. All 15 elements are quite abstract (administrative in the hierarchy we discussed) and all are optional to allow full flexibility. There are many syntaxes for how you record and store the Dublin Core records including in formats such as XML and RDF for machine readability.

Title
Identifier
Subject
Creator (makes the content)
Contributor
Publisher
Date
Description
Language
Type (the nature or genre)
Rights
Source (if derived from another document or record)
Relation
Coverage
Format

You can find some simple examples for each element at https://www.dublincore.org/specifications/dublin-core/usageguide/2000-07-16/generic/ . Some of the elements can be further refined. For example: Date.Created and Date.Modified. More complete documentation can be found at https://www.dublincore.org/resources/userguide/creating_metadata/.

Exercise: Go back to the metadata you created about your favourite book or movie. Can you make this metadata compliant with the Dublin Core standard. You'll need to read the definitions and examples for the 15 elements. If I was to collect all the metadata records from the class, what functionality would be possible if you all use a standard?

Further reading

- Dataset Metadata Checklist - Metadata - UCF Research Guides at University of Central Florida Libraries
- Metadata - Research Data Management - LibGuides at National University of Ireland Galway
- Research Data Alliance, Metadata Standards Directory Working Group
- Digital Curation Centre, Disciplinary Metadata | DCC
- What are Metadata Standards | DCC

How is metadata created?

We've now established that having at least a simple standard can help metadata to be consistent and more useful for finding, understanding and sharing your data.

But, where does metadata come from? In the last exercise you manually created metadata, using the Dublin Core standard, to describe a book or movie. This is an example of manually created, structured metadata where both the metadata attributes and their values are manually entered but there are other sources and methods for creating or capturing metadata.

Let's look at four general sources of metadata:

1. Simple
    - This is metadata that is often generated automatically during the creation of a digital object.
    - Recall in the introduction to this topic, we looked at the embedded metadata in digital photographs.
    - Eg, EXIF, document headers, time stamps
2. Structured
    - This is metadata that adheres to a standard.
    - It may also include manually entered data or conventions.
    - Eg, Dublin Core.
3. Professional
    - Librarians, Archivists, Curators are professional metadata creators!
    - They apply standards and are trained to create consistent, independent and high-quality metadata.
    - Eg, the Library of Congress standards
4. Crowd Sourced
    - Social media content will have examples of simple and structured metadata associated with it but also crowd sourced labels.
    - These may gradually evolve into accepted standard labels from comments or hashtags.
    - Eg., hashtags, comments

These categories apply to all sorts of objects -- both physical and digital -- but what if we consider just datasets? Take a look at these guidelines for creating appropriate metadata to describe a dataset from the University of Central Florida (who have also published a number of datasets for computer vision) - Dataset MetaVdata Checklist - Metadata - UCF Research Guides at University of Central Florida Libraries. Note that many funding agencies will require you to publish a dataset that is created from public funds so understanding the metadata requirements is important.

Discuss: take a look at the recommendations for creating dataset metadata and under point 4, can you recognise the concepts that we have discussed? Are there any suggested metadata attributes that surprise you? Think back to the dataset you looked at from data.gov. Did it have useful and informative metadata?

Hopefully you are starting to realise that creating high-quality metadata can be *expensive*, *time-consuming* and *difficult*. There are some attributes that can be automatically populated but there is still manual work and decisions about the *scope* and *scale* of required metadata.

So how much metadata do we need?

It's important to consider the tradeoffs between:

- organisation (adding, duplicate detection, storage) and
- retrieval (query, search).

You want to be able to fulfill the main requirements for your data or object -- to Find, Locate, Identify, Select, Obtain and/or Navigate -- but not all documents and resources need the same amount of metadata. The principal question to ask is: could someone else understand and use your dataset?

Problems & Challenges

So far I've been very positive and encouraging about metadata, its usefulness and importance, especially for digital datasets and objects. However there are problems and challenges with metadata both in the effort and complication of trying to establish what and how much should be recorded and also in trying to enforce a metadata standard where the information is provided by a variety of other people.

Read this 2001 article by author Cory Doctorow "Metacrap: Putting the torch to seven straw-men of the meta-utopia".

In 2001 there was a particular push for exhaustive, high-quality, machine-readable metadata in the belief that it would create enormous opportunities for new services. In the article, Doctorow looks at seven obstacles that prevent this "meta-utopia":

1. People lie
2. People are lazy
3. People are stupid
4. People delude themselves
5. Schemas aren't neutral
6. Metrics distort or limit
7. There's more than one way!

Some of these ideas will come up again when we look at data cleaning, as metadata created by humans can exhibit a lot of inconsistencies and errors.

Consider: After reading the article do you feel that metadata is no longer useful or important? Do you agree with Doctorow's views on human behaviour? You may find it interesting to look at Doctorow's background and other activities.

References

Baeza-Yates, R. and Ribeiro-Neto, B., 1999. *_Modern information retrieval_* (Vol. 463). New York: ACM press, Addison Wesley.

Gill, T., Gilliland, A.J., Whalen, M. and Woodley, M.S., 2008. *_Introduction to metadata_*. Getty Publications. (Only the chapters on “Metadata and the Web” and “Practical Principals for Metadata Creation and Maintenance”). Available here.

Glushko, B., 2006. 5. *Metadata and Metadata Standards*. Available here.

Preattentive Features

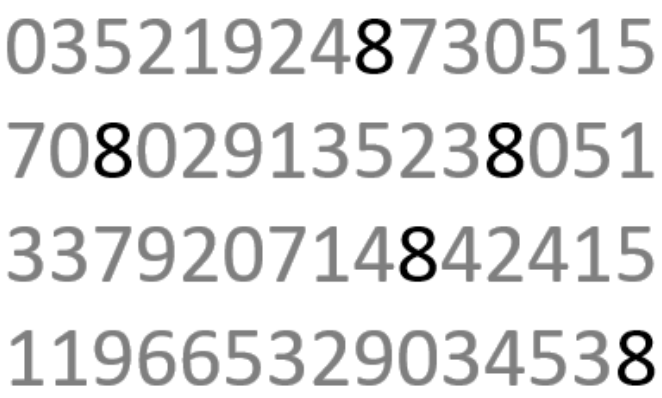
Some visual properties can be detected very very quickly. These are called “preattentive features”.

Preattentive features are generally detected in under 200-250ms, before conscious thought and are immediately perceived. This is useful to direct the viewer’s attention to our main points but can lead to confusion or misleading visualisations if you aren’t thoughtful about their use.

Let’s try an exercise to see how preattentive attributes work. Count the number of 8s in the following set of numbers. Think about how long it takes you.

035219248730515
708029135238051
337920714842415
119665329034538

Now try again:



Notice how the 5 8s leap out at you? It’s much faster now that they are distinct from the other numbers.

This is an example of using a preattentive attribute (intensity of colour) to highlight some data.

There are a number of visual attributes that are preattentive. The figure below (adapted from Healy and Enns (2012), Knaflic (2016) and Few (2004)) shows the most commonly referenced ones for visual communication. You can see a more complete analysis at Healy and Enss, (2012). Each grid has a “target” (the distinct mark) and “distractors” (the other marks).

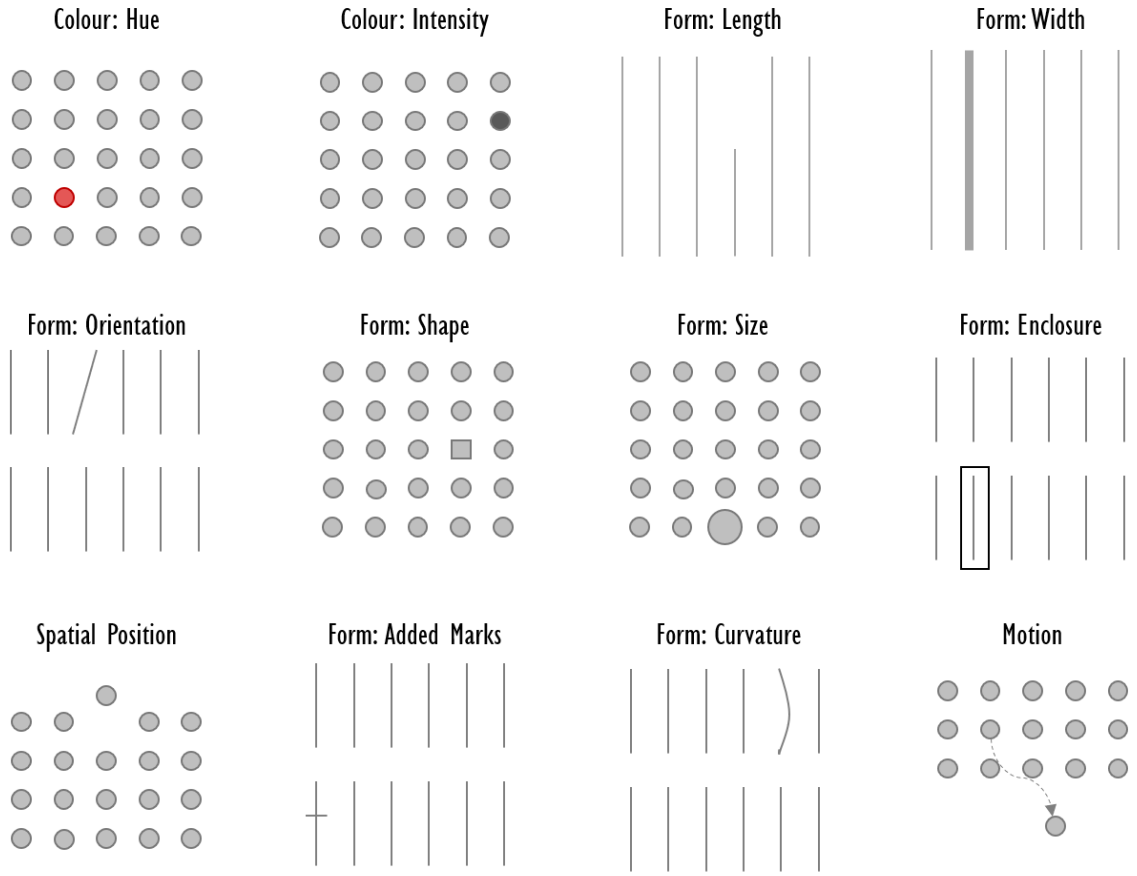


Fig: Common preattentive attributes (from Few (2004), Healy and Enns (2012) and Knaflic (2016)).

There are others. And some variations of those shown here. You could also count “Added Marks” and “Curvature” as a variation of “Shape”. Motion is difficult to represent and isn’t used in a static, 2D image but you’ve seen examples of motion at work in Hans Rosling’s Animated Bubble Charts. A more complete list with links to the original papers where the feature was proven to be preattentive can be found in Healy and Enns (2012). You can also try some of the experiments yourself at that page to see just how fast you detect preattentive features.

What about using some of these in combination? Surely more is better! Let’s see how that works.

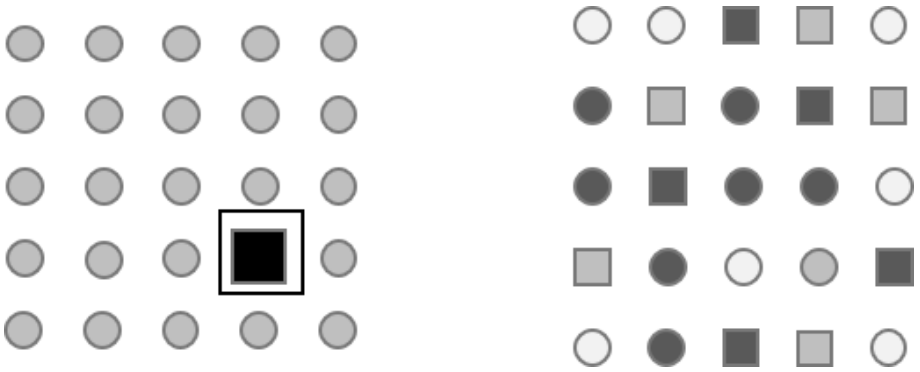
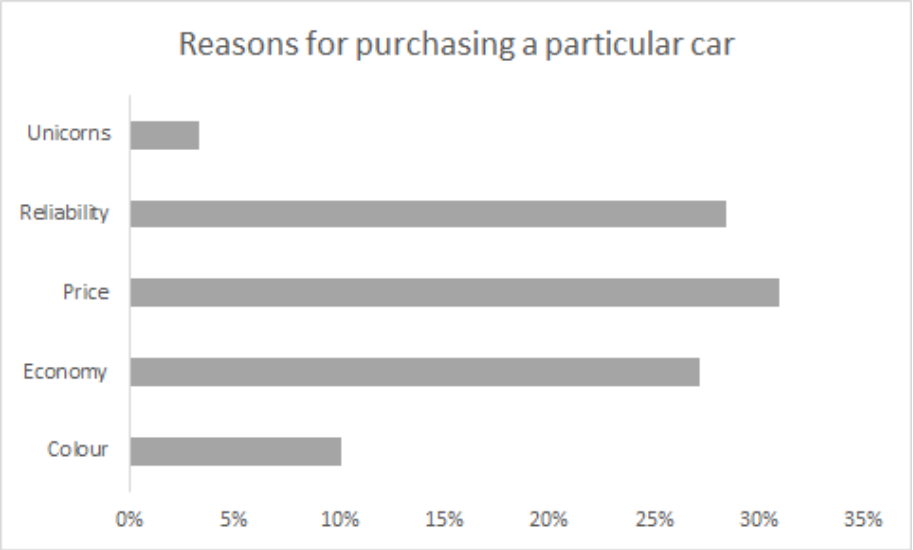


Fig: using multiple preattentive attributes

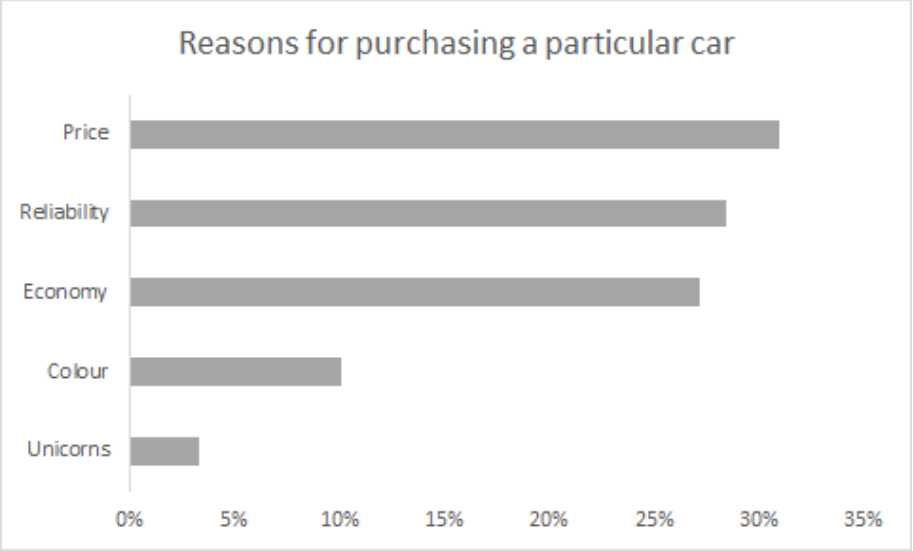
Can you identify which preattentive attributes are used in the image on the Left? Adding more hasn't made us any faster at spotting the target but hasn't made it harder either. However, in the image on the Right it's very difficult to group or count any specific points. You can find the objects that share a single attribute (circle, square, dark, light) but it's harder to separate out combinations like dark circles or light squares. Can you spot the target in the Right image? So it's easy to confuse the preattentive features by overusing them or not making them distinct enough.

Let's look at some examples of how Colour, Form and Spatial Position are used in visualisations starting with this simple bar chart that shows a comparison of the most important (made up!) reasons for purchasing a car.



Form

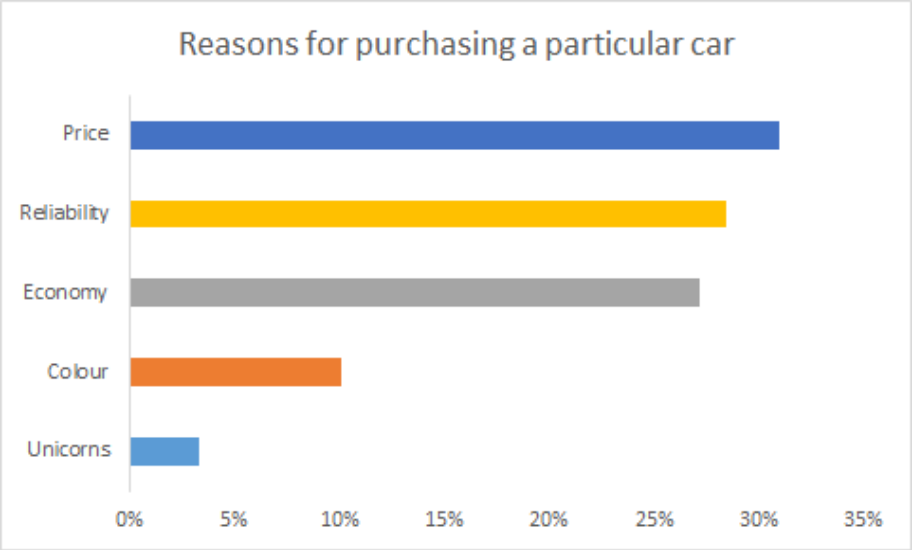
This graph is using length to encode the values and even without any colour or other preattentive attributes you can pretty quickly extract that Price is the feature most people consider important when determining which car to purchase. We can use our knowledge of how people can group visual information to make this more obvious by sorting the values.



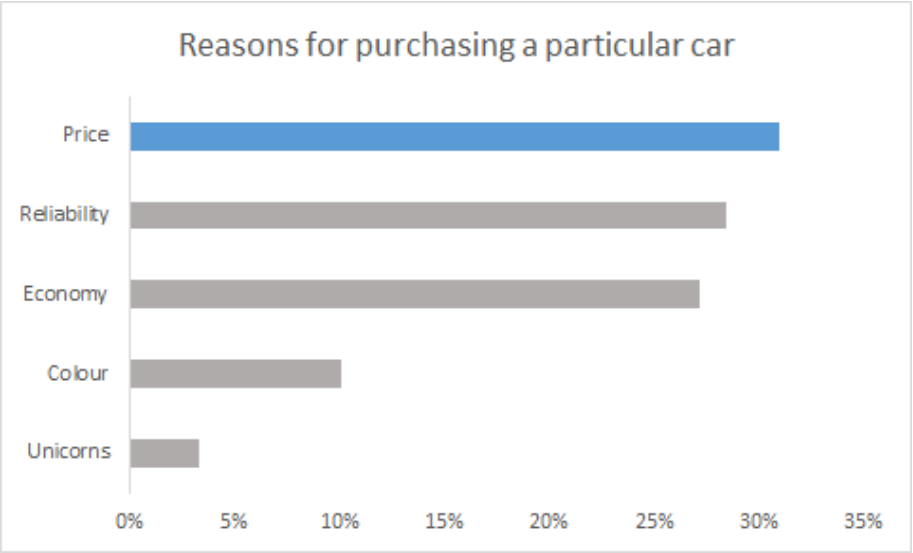
It's now really obvious the Price has the longest bar and is the top feature.

Colour

Now let's try using some colour!



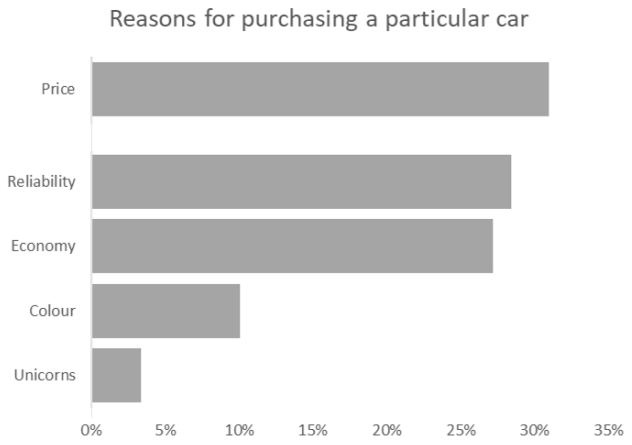
Certainly more colourful but the preattentive feature is overwhelmed. Let's try highlighting with a single "action" colour the story that most people consider Price.



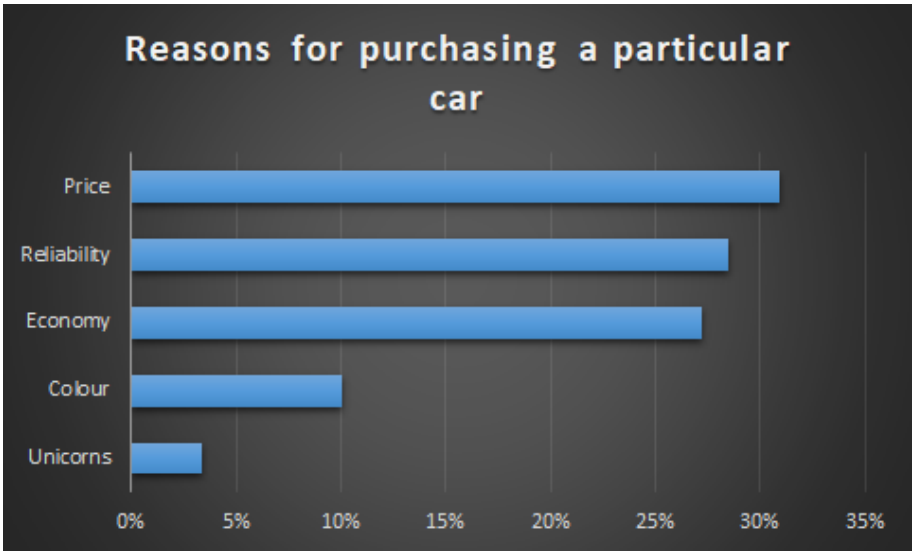
Now we're using the preattentive colour attribute to reinforce the message we're trying to communicate.

Spatial position

Spatial position within this graph example isn't so effective but let's see how we could use it.



Spatial position highlighting is better seen by looking at the white space (negative space) used to surround the graph. Below is an example where an unnecessarily dark background is distracting.



Summary

The target -- the mark or relationship that you want to highlight as part of your message -- must stand out in a simple dimension like colour, orientation, size, position or through motion if that's available. Overusing these attributes can lead to confusion and distraction so use wisely.

We will also look at the Gestalt theory of psychology that looks at how we perceive pattern, form and organisation. Combining an understanding of preattentive features with knowledge of Gestalt principles will help you **avoid distraction** while directing the searchlight of **attention to the key points** of your message.

References

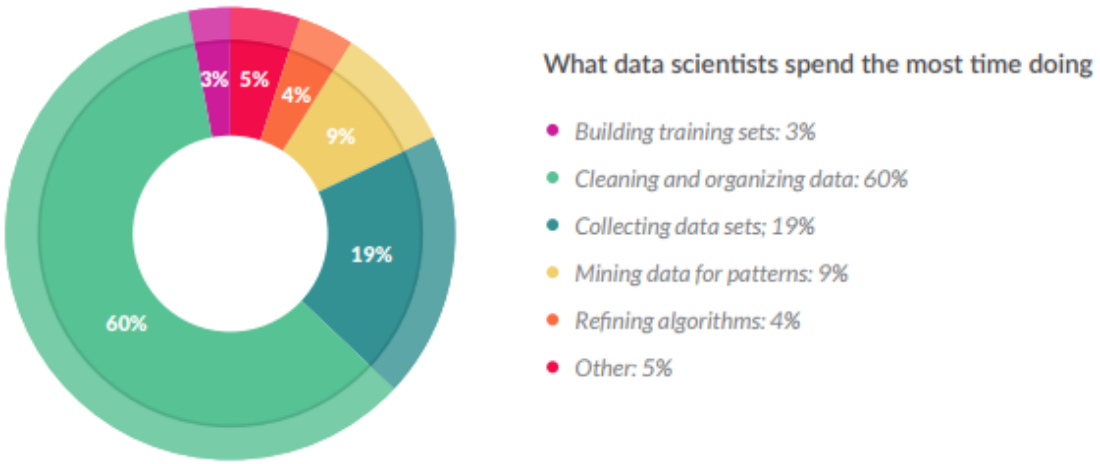
Stephen Few, “Show me the numbers”, Chapter 5
Healey, C. G. and Enns, J. T. "Attention and Visual Memory in Visualization and Computer Graphics." IEEE Transactions on Visualization and Computer Graphics 18, 7, (2012), 1170–1188. Web page with content at Perception in Visualization.
Cole Nussbaumer Knaflic, “Storytelling with Data” (2015), Chapter 4

A Visualisation Process

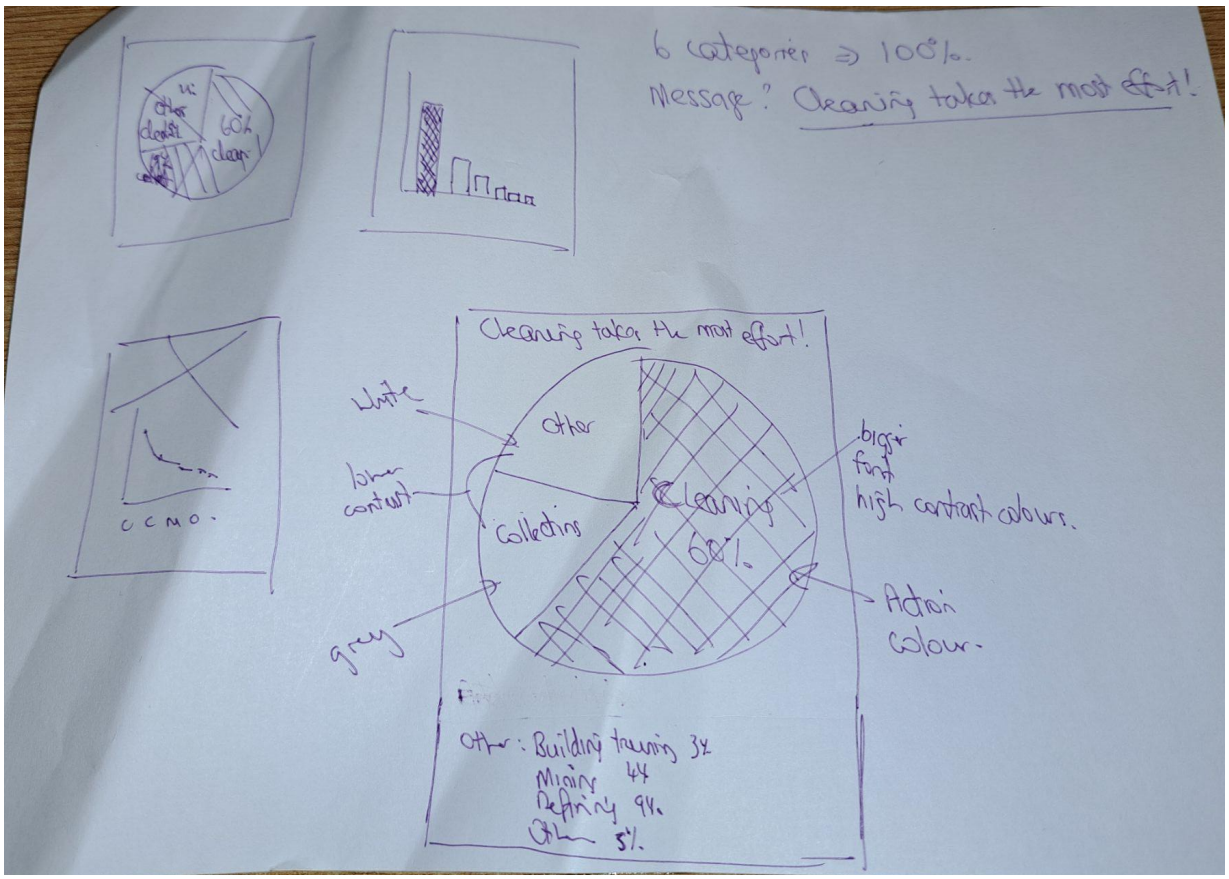
Here is an outline of a visualisation design process for creating an effective data-driven visualisation. Refer to the information from previous weeks to understand the communication process (message, receiver); choosing charts and attention elements.

1. What is your data? What is your message or story? Why are you visualising?
2. Choose an appropriate chart
3. Sketch what it might look like. Identify aspects you want to draw attention to.
4. Make your graph
5. What elements of the graph could be removed? (Data-to-pixel ratio)
6. What attentive elements are you using? Is the attention drawn to your message?
7. Consider design rules (e.g., good contrast, layout) or branding requirements

Example Sketch



http://visit.crowdflower.com/rs/416-ZBE-142/images/CrowdFlower_DataScienceReport_2016.pdf



To note about sketches:

- Thumbnails to get impression
- Don’t try to be very accurate but do watch for scale issues
- Don’t use rules, compass, graph paper → should be rough
- Label or annotate with your ideas or notes to consider for colours or other attentive features.

Gestalt Theory

The Gestalt Theory of Visual Perception (Max Westheimer, Kurt Koffka and Wolfgang Kohler (1912)) is based on the concept that we view scenes as a combination of the parts and create order from the inputs. The *parts* that make up the image of a tree (grass, branches, trunk, leaves) are less noticeable than the concept of “tree”.

If you think back to how we use visual attributes to encode data, you may recall that of the encoding attributes *Position* was the top for understandability regardless of the data type. This is connected to Gestalt principles and how we group and perceive the world.

Watch the video (~5mins) below introducing Gestalt principles and their influence on design.

https://youtu.be/LlzuJqZ797U

One concept in Gestalt theory that you may have seen before is the idea of *Figure and Ground* where figure is the foreground object and ground is the surroundings (or background). The viewer tries to divide the image into these components and you get the interesting visual effect seen in the Rubin Vase image (below) or the picture at the start of this article. Do you see two faces or a vase?

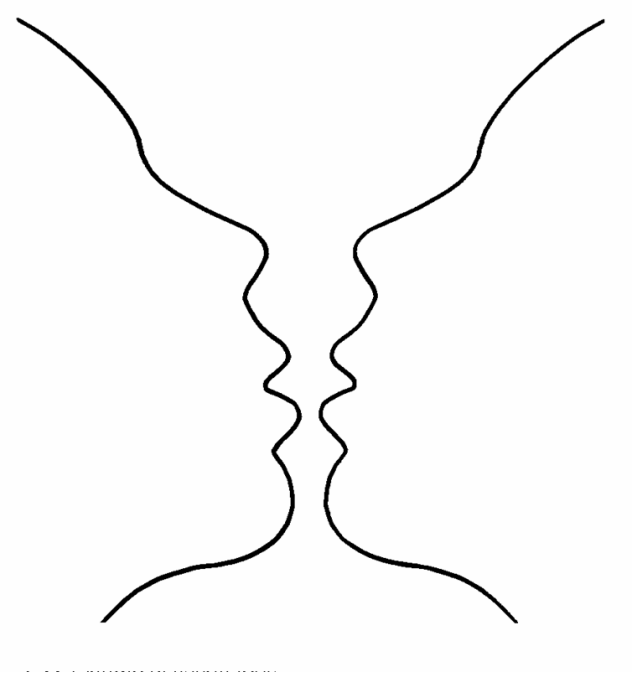


Fig: an example of a Rubin Vase image

Paying attention to figure and ground means making sure that the contrast, grouping and borders of your graph are sufficiently contrasted from the background to make it easier for the viewer to distinguish figure and ground without weird visual effects.

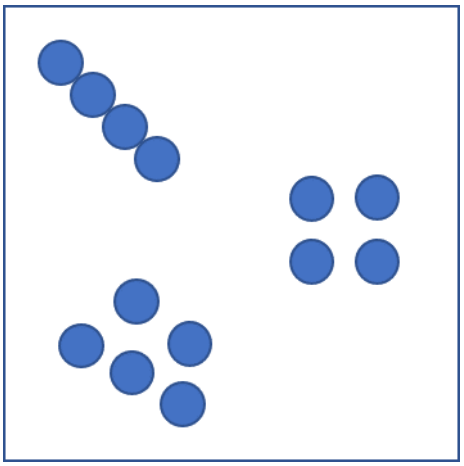
We will look at six principles here that can be applied to understand how the viewer is interpreting your visualisation, grouping elements and creating patterns.

1. Proximity
2. Similarity
3. Enclosure
4. Closure
5. Continuity
6. Connection

Proximity

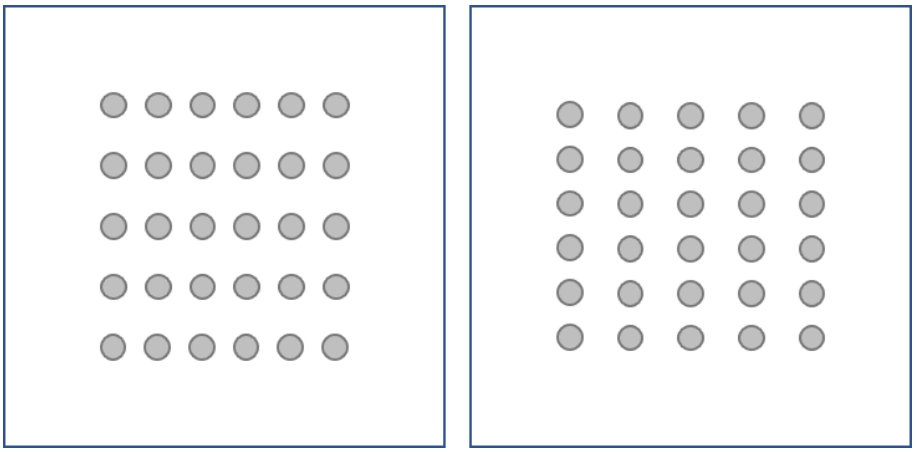
Description: Elements placed together are seen as a whole

Illustration



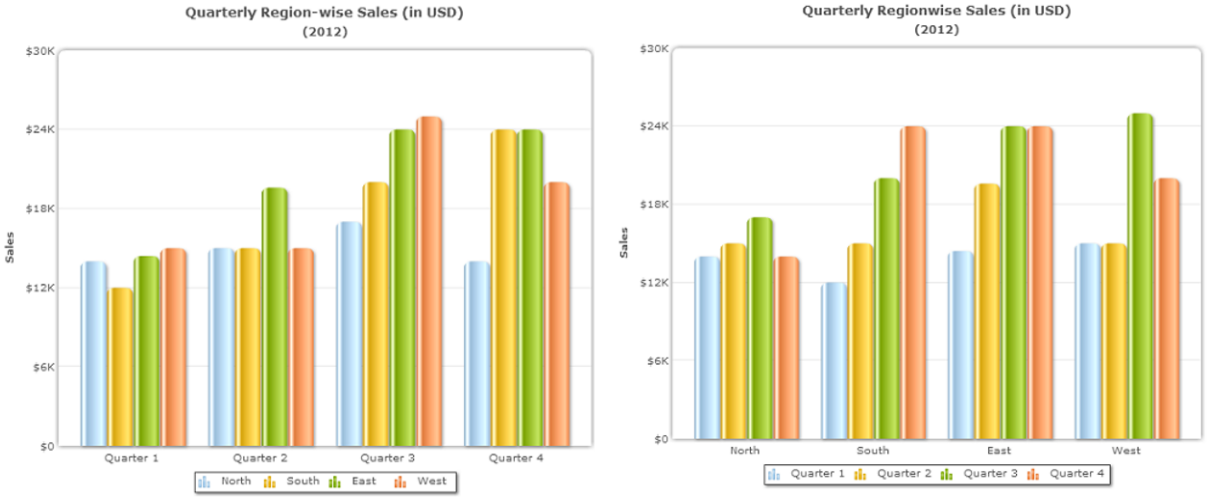
You perceive this as 3 objects rather than 13 circles due to the proximity of the groupings.

Leveraging



In the image above you distinctly see rows on the left and columns on the right only due to adjustment of the spacing (proximity). This is used in designing tables and in clustered bar charts. The viewer will group the elements in close proximity so make sure this is the attribute that you want the viewer to group.

Example

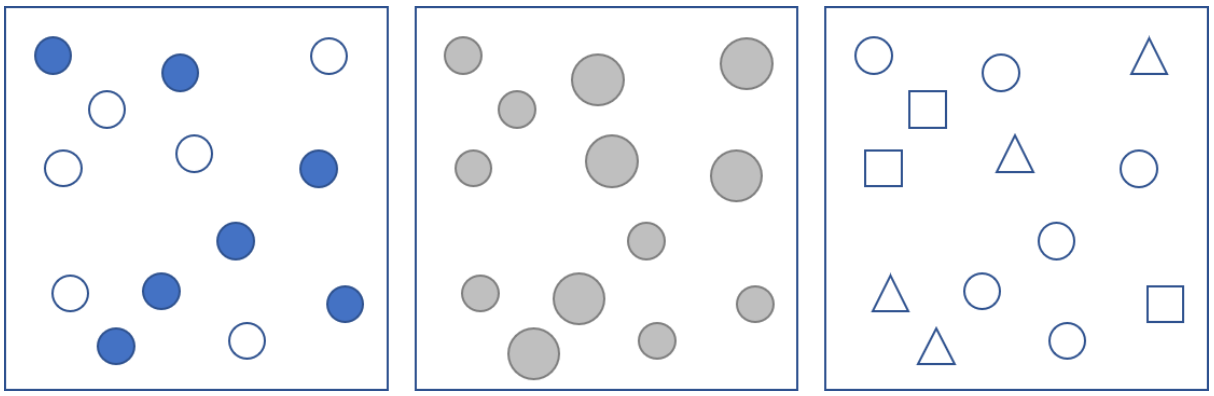


In the example graphs above (from https://www.webfx.com/blog/web-design/data-visualization-gestalt-laws/), the first grouping is based on the financial quarter but this makes it hard to compare performance in the Northern region.Grouping instead by region (graph on the Right) makes it easy to compare this.

Similarity

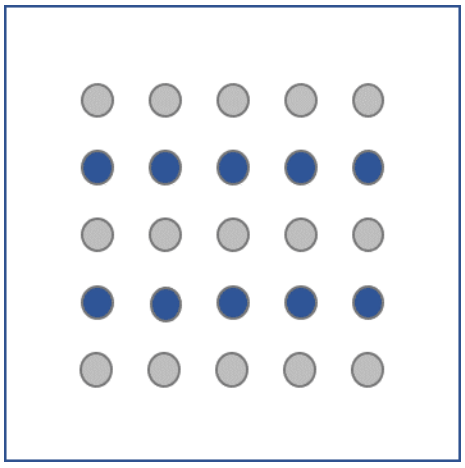
Description: Elements that share similar attributes (e.g., colour, size, shape, orientation) are seen as a whole

Illustration



You see different groupings based on the similarity of the objects. Left: colour shows 2; Middle: size shows 2 and Right: shape shows 3.

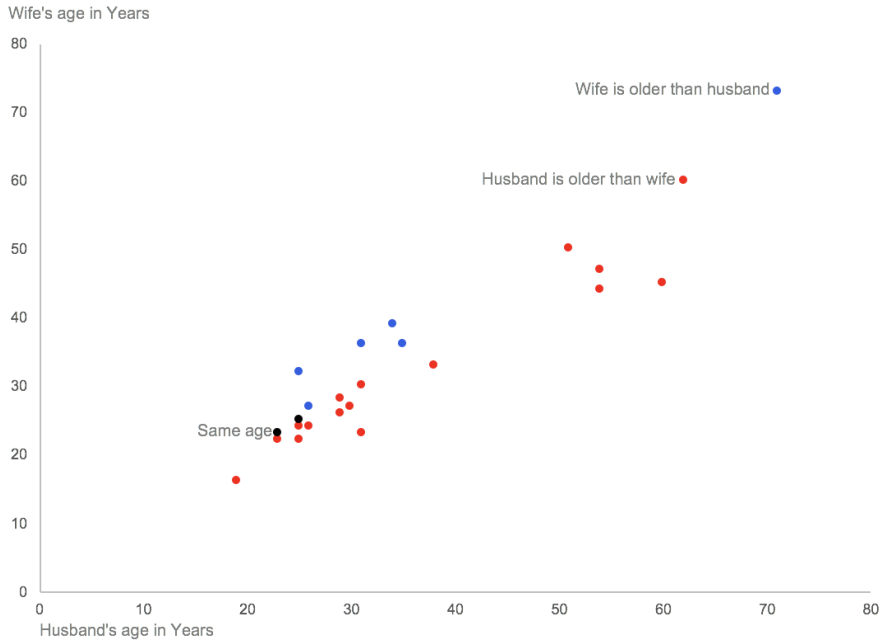
Leveraging



You see rows because of the colour similarity though the spacing here is consistent.

You can “break” the principle of similarity to highlight an anomaly. The eye is drawn to the figure on the Right because it is dissimilar to the others. This is how preattentive attributes use gestalt similarity (or lack of it) to highlight a point.

Example



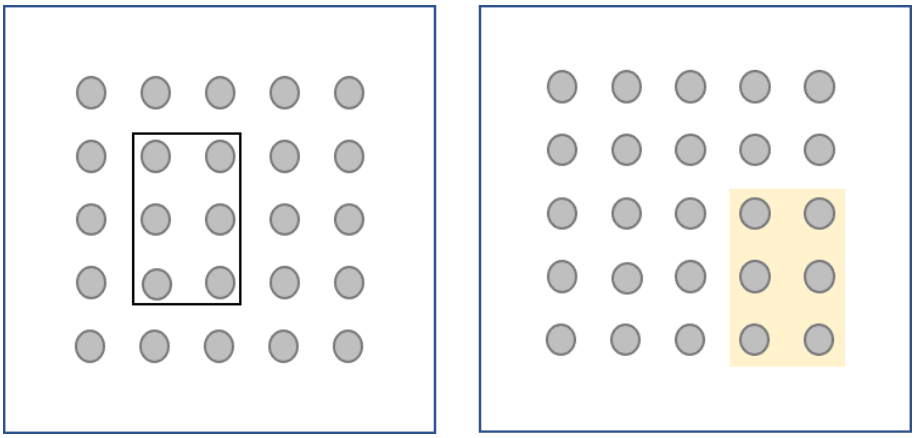
The use of colour to indicate the grouping allows us to see the different categories (red, blue and black) even though they are spatially distributed.
(from: http://daydreamingnumbers.com/concepts/gestalt-laws-data-visualization/)

Note the correlation of this use of colour or similarity: if there are too many different colours with no clear meaning then it adds cognitive complexity and increases confusion.

Enclosure

Description: Objects that appear to have a boundary around them are perceived as a group

Illustration

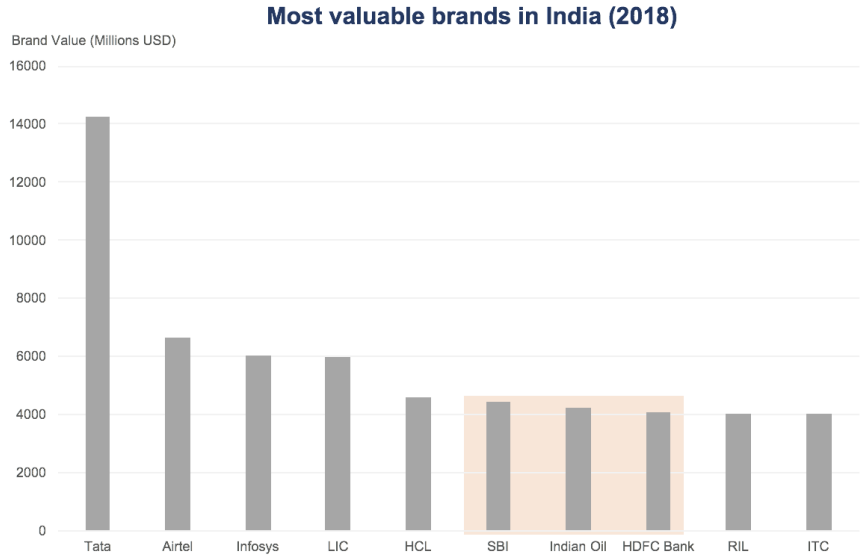


The two sets of circles are identical but the enclosure elements (box and subtle highlighting) lead us to group them differently.

Leveraging

Enclosure allows us to easily create a grouping of related marks or to highlight sections of our data. Borders and background fill colours are commonly used in tables and graphs. Note that they don't need to be (and shouldn't) be strong, overwhelming attributes to create the grouping. Subtle indicators are all that's needed. In fact, as you'll see in the next principle (Closure) we are very good at implying a complete boundary even where it's not there.

Example

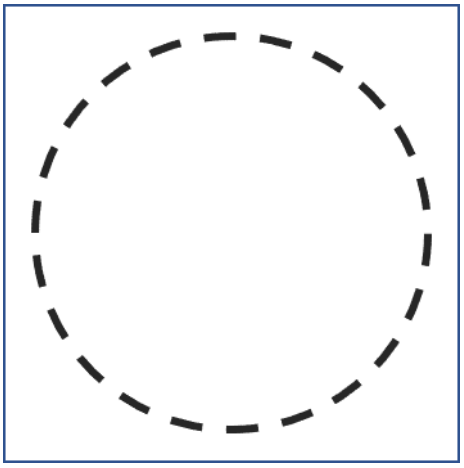


Use enclosure to highlight and group a set of points like the bars shown above. This creates a visual distinction.
(from: http://daydreamingnumbers.com/concepts/gestalt-laws-data-visualization/)

Closure

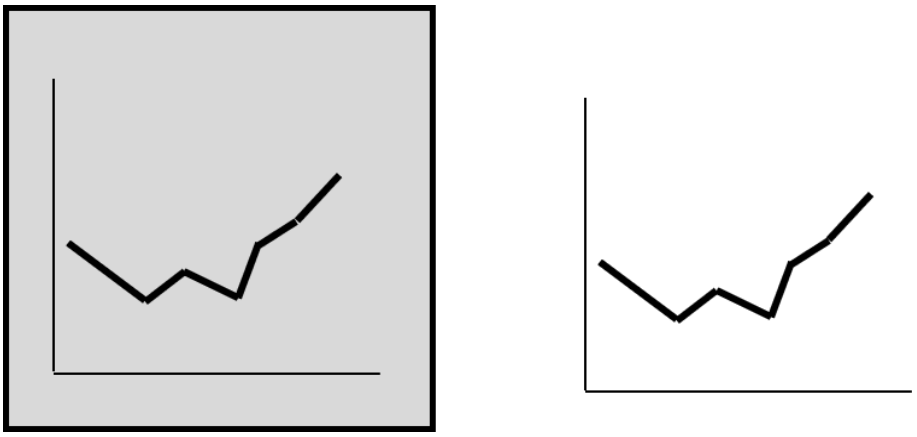
Description: Open structures are perceived as closed and complete wherever possible. Note that people are very good at “joining the dots”!

Illustration



You see the dashed lines as a complete circle as our eyes fill in the gap with the expected relationship.

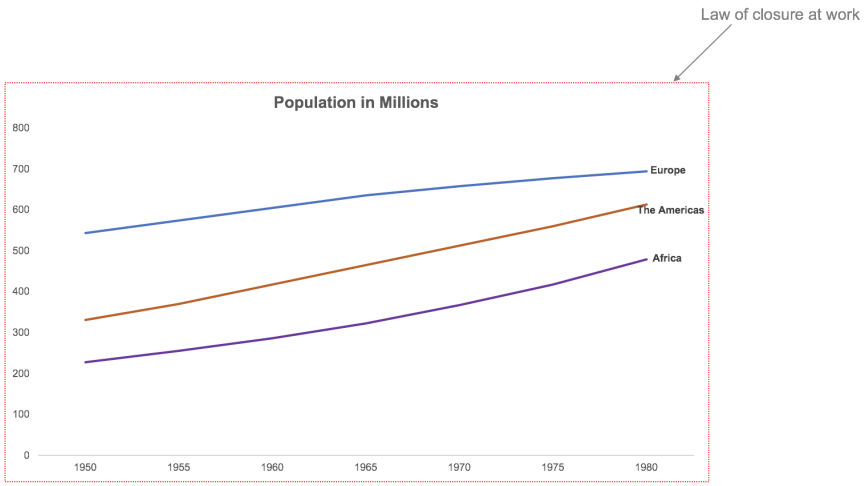
Leveraging



The image above shows how even without the explicit background and border enclose we are still able to imply the enclosure of the graph as a grouped object. While including the background and border may be a default option in many graphing applications, removing this lets us reduce clutter in our graphs and improve readability. We’ll look at this further in the next topic.

We should also be careful about gaps in our visualisations, perhaps due to missing data. The viewer may inadvertently “fill in” the gap where we don’t want them to.

Example

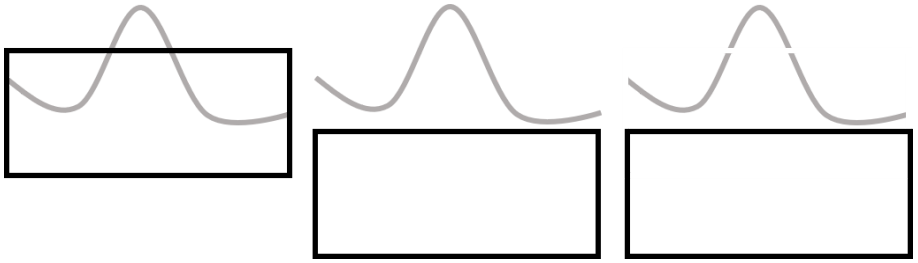


We view the graph above as a whole even without a distinct border. (from: http://daydreamingnumbers.com/concepts/gestalt-laws-data-visualization/)

Continuity

Description: Elements that appear to be sequential and/or forming continuous lines are seen as a grouping. We tend to imply this continuity to the strongest degree possible, grouping objects into the fewest options.

Illustration



We assume that the image on the Left is made of a rectangle and a curved line (Middle) and don't tend to see it as a rectangle and three line fragments (Right).

Leveraging

Similarly to removing borders and backgrounds by exploiting the principle of closure, we can use this to remove the explicit axis line and the viewer will fill it in as shown below.

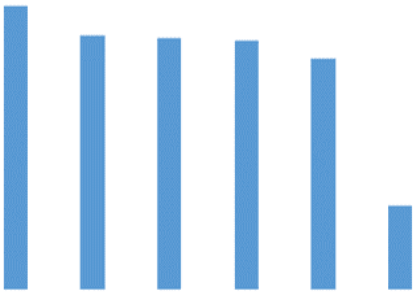
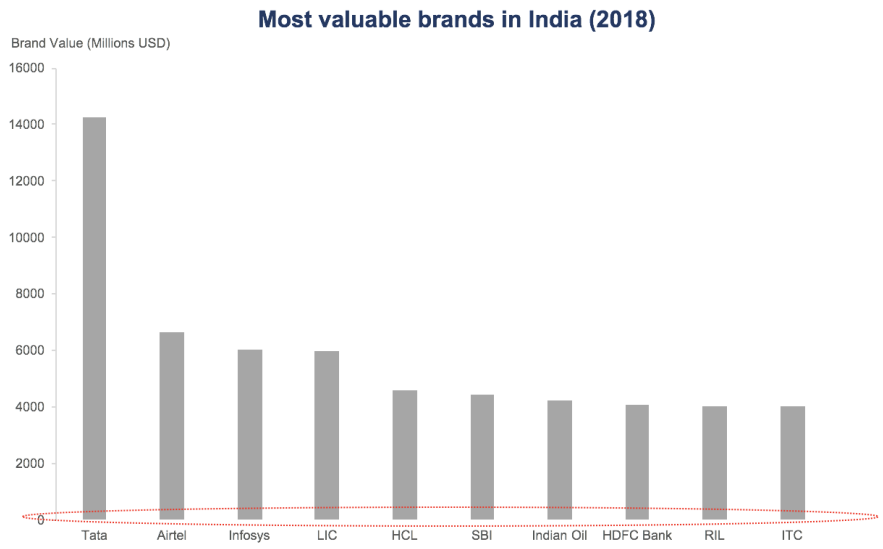


Fig: Still recognisable as a (badly labelled!) graph even without the x-axis line.

The image above is recognisable as a football (soccer ball) even without the explicit joining. The continuity of the black sections implies the rounded and familiar shape and the principle of closure invites the viewer to join the sections.

Example

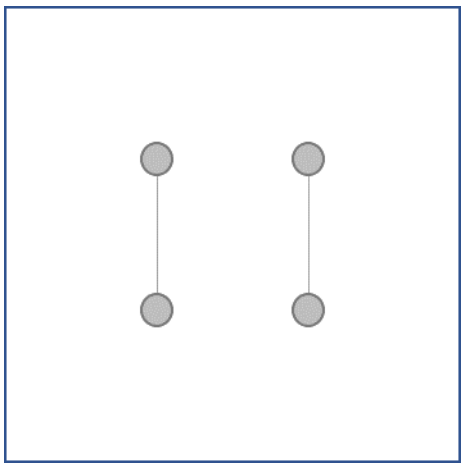


Note the missing x-axis line
(from: http://daydreamingnumbers.com/concepts/gestalt-laws-data-visualization/)

Connection

Description: Objects that are connected (e.g., by a line) are perceived as a group. Similar to enclosure we can draw an explicit link between objects using lines to connect them.

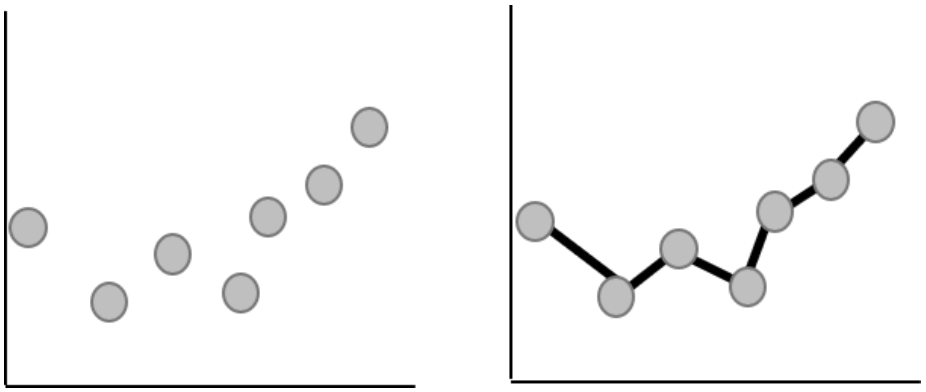
Illustration



We see the image above as two groups even though the dots are identical and identically spaced.

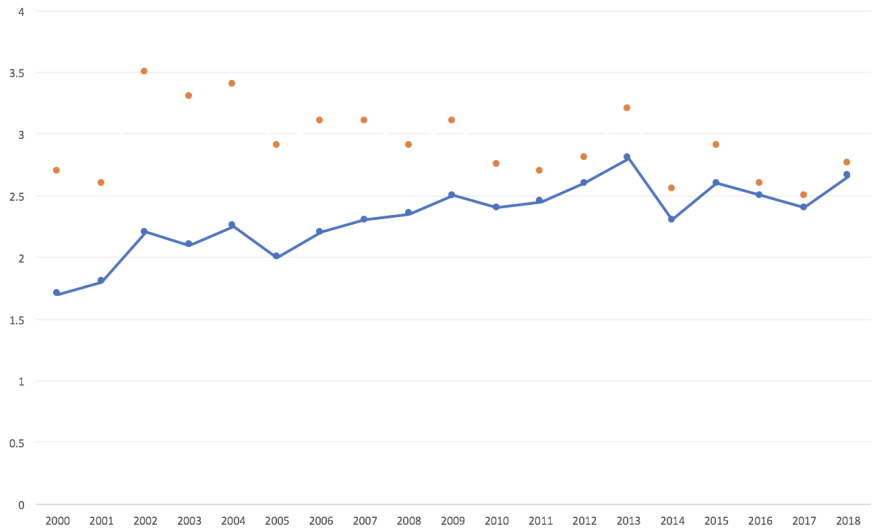
Leveraging

We commonly use this principle in line graphs which are a series of connected points.



Adding the line not only connects the dots but adds information about the overall shape of the data.

Example



The blue line connects those points into a single object while the orange dots appear to be scattered and unrelated. (from: http://daydreamingnumbers.com/concepts/gestalt-laws-data-visualization/)

Summary -- why does this matter?

We've had a few topics so far that deal with how we see. Why does this matter? In ensuring that our visual communication is as effective as possible, you should keep the following principles in mind:

- Making most of visual cues (preattentive attributes)
- Not overloading the visual information channel (cognitive processing load)
- Not excluding or miscommunicating with your audience (colour, visibility and accessibility)
- The problems with simulating effective depth, motion and using 3D effects in a 2D visualisation (depth perception)
- Engaging attention (preattentive attributes and gestalt principles)
- Not diluting your message (distraction, gestalt principles)

References

Chapter 4, Storytelling with Data

Chapter 5, Show me the numbers

Healey, C. G. and Enns, J. T. "Attention and Visual Memory in Visualization and Computer Graphics." IEEE Transactions on Visualization and Computer Graphics 18, 7, (2012), 1170–1188. Web page with content at Perception in Visualization.

Gestalt Theory of Visual Perception

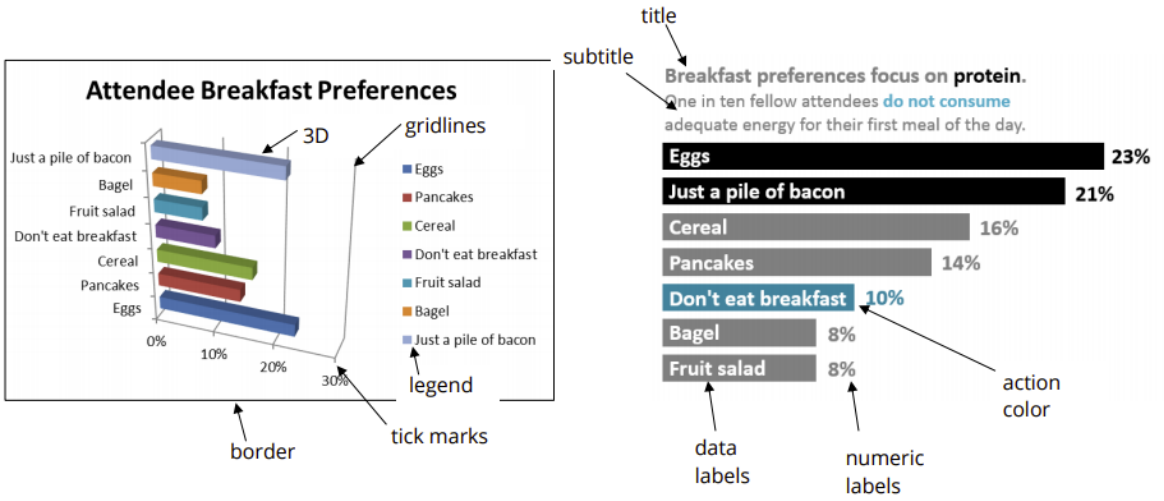
Nice animations showing the different principles: Gestalt Principles for Data Visualization: Similarity, Proximity & Enclosure

Exercise: Critiquing Graphs and Visualisations

You can share your results and comments at https://forms.gle/3DeUyJZknHtaC63m7

1. **Apply Checklist from Stephanie Evergreen**

http://stephanieevergreen.com/updated-data-visualization-checklist/

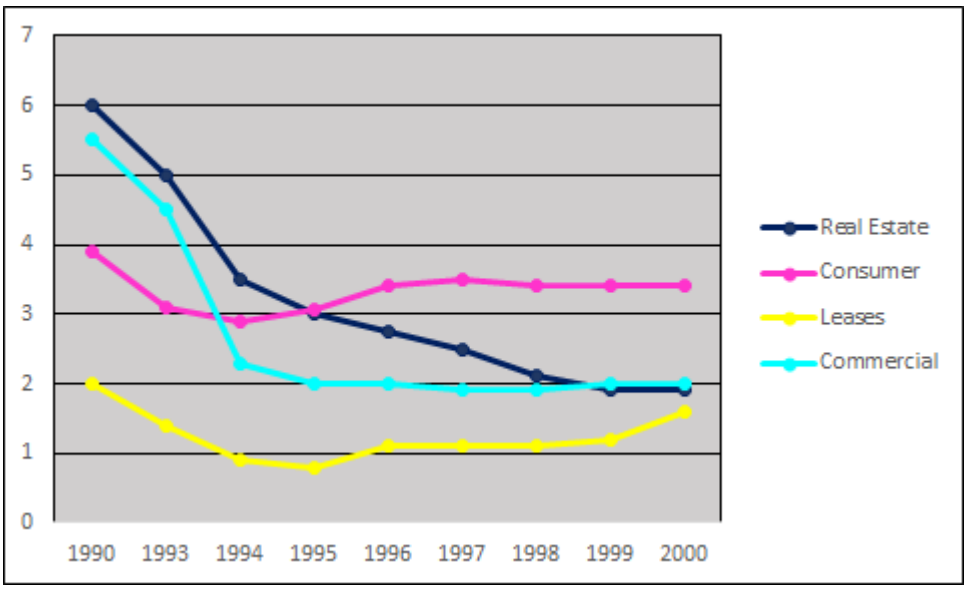


Use the evaluation checklist from Stephanie Evergreen to assign a number to the following two graphs. Note that there is some judgement here so you might not get the same number as someone else.

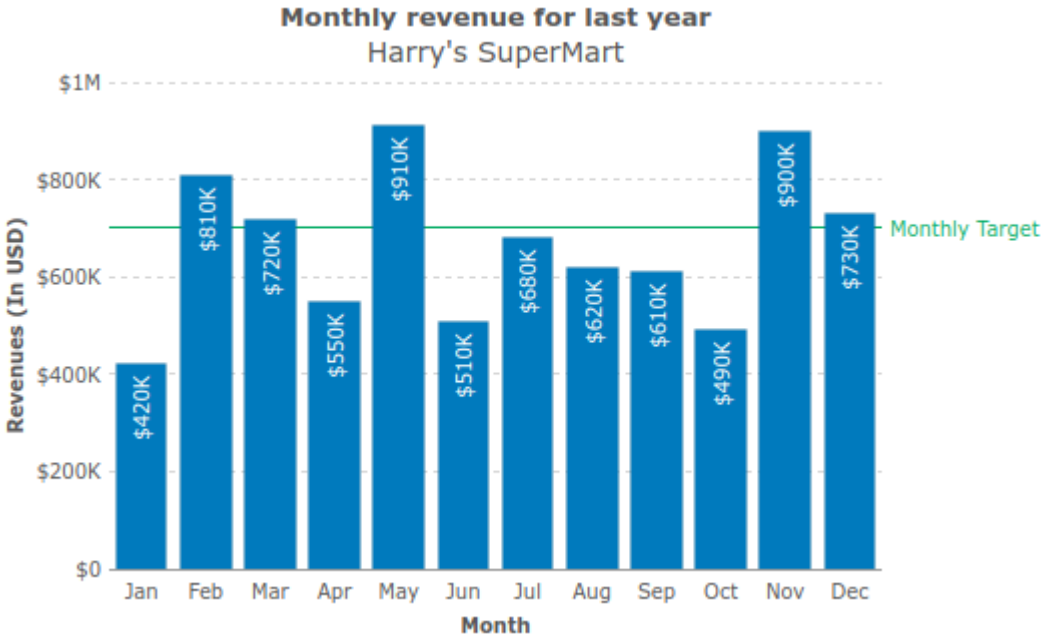
You should give your score as the percentage where the total is all the points where you haven't decided the guideline is n/a (not applicable).

Enter your scores as a percentage between 0 & 100 in https://forms.gle/3DeUyJZknHtaC63m7.

Graph A:



Graph B:



2. **Analyse and suggest specific improvements for the following graph**

The following example is taken from Knaflic, Cole. Storytelling With Data: Let's Practice! Wiley, © 2019 available from http://www.storytellingwithdata.com/letspractice/downloads.

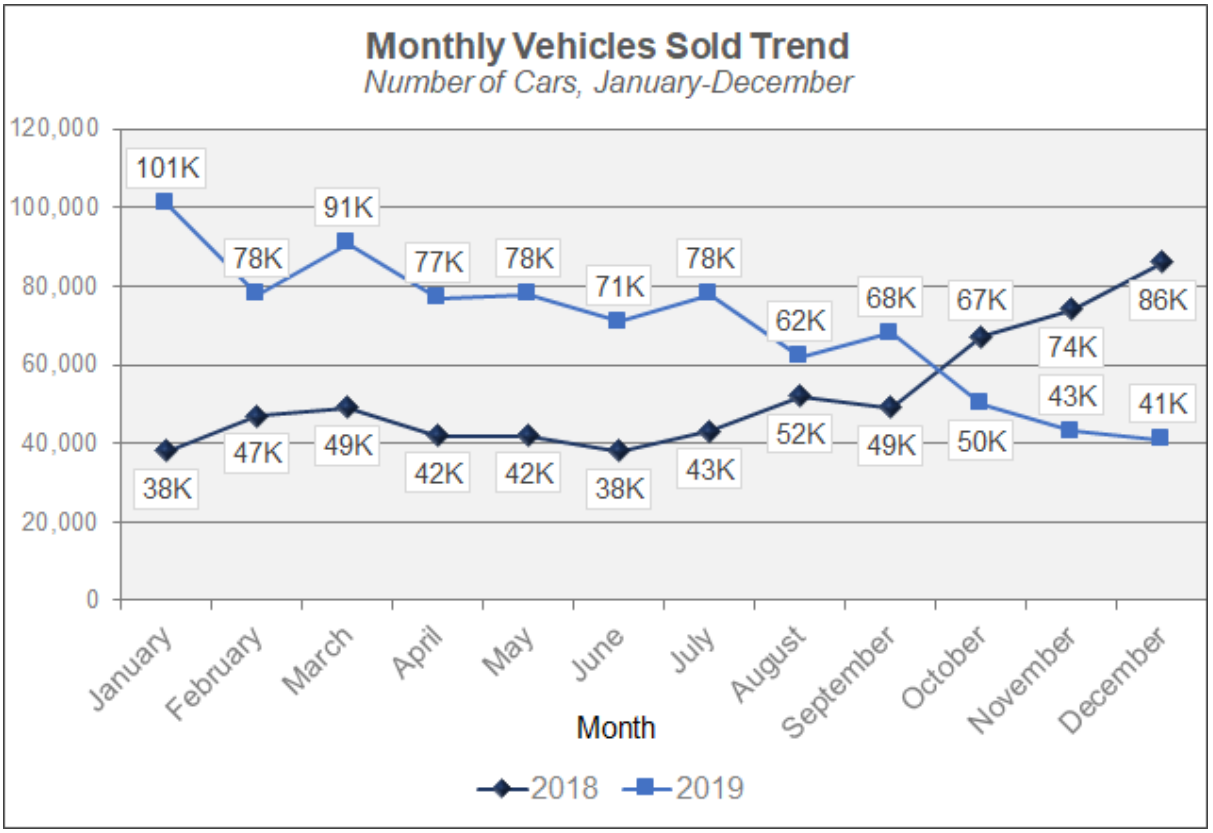


Fig: Exercise 3.8 from Let's Practise

List all the ways you could remove clutter from the original graph and put these into the form at https://forms.gle/3DeUyJZknHtaC63m7.

Try to edit or recreate the graph (you can change graph type) to produce a more effective, clutter free version. You can download the data and original graph (Excel format) from https://drive.google.com/open?id=1iEkFh-5h5tDoLWvJDIds4z9GqK7uzfSC