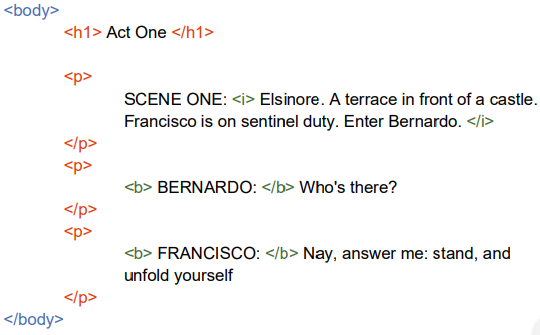
# Data Formats

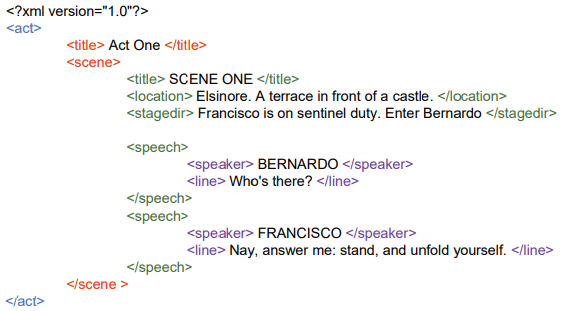
**Regular expressions** are specified patterns in text. Useful for filtering out strings in texts.

|  |  |
| --- | --- |
| . | Matches any character |
| ^ | Matches the start of a string |
| $ | Matches the end of a string |
| \* | Has zero or more repetitions |
| + | Has at least one repetition |
| | | OR operator – used with parentheses () |
| [] | A set of characters – [abcd] or [a-zA-Z] |

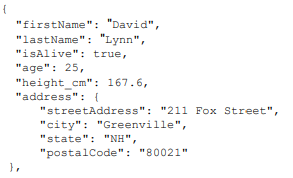
**HTML** (Hypertext Markup Language) is designed for pure presentation. Elements are delineated by tags which correspond to logical units – such as headings, paragraphs, tables, etc. They are not case sensitive, and HTML can be inconsistently applied - it will format it regardless.

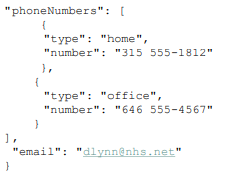


**XML** (eXtensible Markup Language) must begin with a declaration <?xml version=”1.0”?> and contain at least one single root element. Tags are case sensitive and must be properly applied. Attributes inside elements must be in quotes (Example: <person title=”Mr”>Name </person>). XML also allows complex schema definitions (using regular expressions) for formal validation and forces you to consider the data design much closer.



**JSON** (JavaScript Object Notation) is simpler and more compact compared to XML. It is easier to parse and can be read directly through JavaScript. It is essentially a dictionary type which can store: numbers, strings, bool, arrays, objects and NULL. JSON is designed for speed and efficiency – being widely used in noSQL databases.

A representation of a dictionary entry in JSON

  
compared to one in XML.



# Data Cleaning: Missing Values and Outliers Detection

Measures of data quality: accuracy, completeness, consistency, timeliness, believability and interpretability.

|  |  |
| --- | --- |
| Noisy Data | Incorrectly split strings, truncated fields, strings mixed with integers |
| Inconsistent Data | Different naming representations, different DateTime formats |
| Intentionally Disguised Data | Unusual or suspicious values that should not be part of the dataset |
| Incomplete/Missing Data | Entries lacking values, NULL/NaN, empty entries |

Missing values – Completely at Random:

* Missing data on a variable is unrelated to any other measured variable (on context of dataset)
* Example: Test results but did not take it and thus could not share

Missing values – Not at Random:

* Missing data on a variable is related to the values of the data itself (on context of dataset)
* Example: Test results but did bad and did not want to share

So how do we deal with missing data?

1. Delete all instances with a missing value
   * Easier to analyse the new complete data
   * May produce bias/skew on analysis if the sample size is small
2. Manually correct missing values
   * Time consuming
   * Can you even correct the missing values?
3. Imputation - Find a substitute value to replace the missing ones
   * Fill with 0:
     + Won’t break applications
     + Limited utility for analysis
   * Fill with mean:
     + Good for supervised classification
     + Applicable separately to each attribute
     + Reduces variance of the feature resulting in an incorrect view of the distribution of the feature
     + Relationships to other features change
   * Fill with median:
     + May be a better estimator if the distribution is skewed

**Outliers** are data objects that **deviate significantly** from the general distribution of data. They can be different from noisy data (a random error or variance in a measured variable and should be removed before outlier detection) and is closer to a completely unexpected value from the distribution of the data. To visually detect outliers, a boxplot or histogram will show it within reason.

A **five-number summary** of the data will also be sufficient. Applications of outlier detection include: credit card fraud, medical analysis, sports, telecom fraud, etc.

Types of Outliers

1. Global Outliers (Point Anomaly)
   * An object is a point anomaly if it **significantly** deviates from the rest of the data set
2. Contextual Outliers (Conditional Outlier)
   * An object is a conditional outlier if it deviates **significantly** **based on a selected context**
   * An example may be: “Is 5 degrees an outlier” – depends if its Winter or Summer
   * Attributes of data should then be divided into two groups:
     + Contextual Attributes that define context (time, location, etc.)
     + Behavioural Attributes that define the characteristics of the object used in outlier evaluation (temperature)

# Recommender Systems

How do recommender systems work?

* Each user has a personalised profile
* Users then use the platform and watch/rate items:
  + Explicitly by giving a score
  + Implicitly by web usage mining
* The recommender system then uses **collaborative filtering** to make predictions about a user’s missing data, according to the behaviour of many other users
  + Looks at users with similar behaviour
  + Looks at the user’s history and past actions

Collaborative filtering approaches:

1. User based methods by identifying like-minded users
2. Item based methods by identifying similar items that the user has viewed
3. Model the methods with a simple matrix – solve an optimization problem and identify latent factors

Similarity measures (User to User or Item to Item):

1. Compute the squared Euclidean distance between the mean value of A’s missing values and the mean value of B’s missing values
2. Compute the squared Euclidean distance, summing the pairs from A and B that do not have missing values. Scale this result according to the number of total attributes.
3. Use correlation (Pearson’s correlation method using covariance and variance of A and B)
4. Cosine similarity by computing the angle between two user profile vectors

# Advanced Data Visualization

Parallel Coordinates:

* Used for plotting feature values of high-dimensional data
* The feature values of each object are plotted as a point on each coordinate axis connected by a line
* Objects are represented as a line which are a distinct class of object grouped together for some features
* The ordering of attributes is important to see such groupings
* May be a good idea to scale all features into [0,1] via pre-processing

# Clustering and Clustering Visualisation

A good clustering algorithm will produce a higher quality cluster – objects within the same cluster are **close together** whilst objects in a different cluster will be **far apart**. Clustering algorithms are typically **distance based** that represent each object as a vector which can be used to compute Euclidean distances. Each object may only be assigned to one cluster. These clusters are then summarised by its **centroid** – the average of all its objects.

K-Means algorithm:

* Given a parameter *k*, the *k-means* algorithm is implemented in four steps

1. Select *k* seed points as the initial cluster centre
2. **Assign** each object to the cluster with the nearest seed point
3. **Compute** the new seed points as the centroids of the clusters of the current partitioning
4. Repeat until the assignments stop changing

* The *k-means* will typically choose initial seed points randomly and can simulate numerous values to produce different results
* The closeness is measured by Euclidean distance
* The algorithm can be shown to converge to a local max which usually doesn’t require many iterations

An outlier is **expected** to be far away from any groups of normal objects. Each object is associated with one cluster and its outlier score is computed by the distance from its centroid.

Dissimilarity Matrix D:

* We can visualise a dissimilarity matrix as a **heat map**
* The diagonal of matrix D is all zero since it is a triangular matrix
* A better visualisation can be implemented by ordering the objects

VAT (Visual Assessment for clustering Tendency):

* A good VAT visualisation will suggest both the number and size of clusters
* Diagonal dark blocks appear only when tight groups exist within the data

# Hierarchical Clustering and Dimension Reduction

Hierarchical clustering:

* Produces a set of nested clusters which are organised as a hierarchical tree
* Does not need a given number of clusters
* May correspond to meaningful taxonomies
* Two main methods of hierarchical clustering:

1. Agglomerative:
   * + Starts with the points as individual clusters
     + Merge the closest pair of clusters until one cluster is left
2. Divisive:
   * + Start with one massive cluster
     + Split the cluster until each cluster contains a single point

How do we define Inter-Cluster Similarity?

* We define the similarity to be the **minimum distance** between the clusters. This is known as **single linkage**
* **MIN** (Single Linkage) that connects the closest object of each cluster
* **MAX** (Single Linkage) that connects the furthest object of each cluster
* The **similarity** of two clusters is based on the two most similar points in the different cluster

Max complete linkage is **less susceptible to noise and outliers** whilst Min complete linkage **tends to break large clusters**

# Assessing Correlations

Correlation is used to detect pairs of variables that **may** have some **relationship** (correlation does not imply causality)

Why is correlation important?

* Discover relationships and causality, “A causes B”
* Need to use **feature ranking** to select the best features for building better predictive models
* A good feature is a feature that has high correlation with the outcome one is trying to predict

# Mutual Information

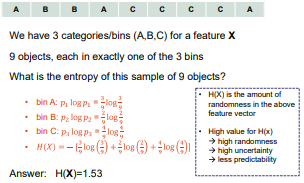
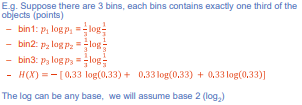
Mutual Information is a correlation measure that can detect **non-linear relationships** and operates with **discrete features**.

Variable discretization:

* **Domain knowledge**: Assign the thresholds manually
* **Equal-Width bins**: Divide the range of continuous feature into equal length intervals
* **Equal Frequency bins**: Divide the range of continuous feature into equal frequency interval bins – all bins will have the same number of objects

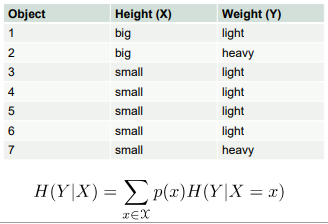
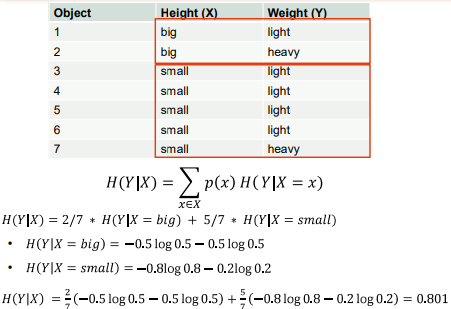
Entropy:

* A measure used to assess the amount of **uncertainty** in an outcome
* Let X be the number of categories and p be the proportion of the i’th bin, then we can find entropy through



Conditional Entropy:

* Measures how much information needed to **describe Y,** given that **outcome X is known**.

Mutual Information:

* Let X and Y be features (columns) in a given dataset, the Mutual Information is a measure of correlation that tells us:
  + The amount of information on X we gain by knowing Y
  + The amount of information on Y we gain by knowing X
  + The larger the MI(X,Y), the more they are dependent on each other
* The normalised mutual information can be calculated by dividing the Mutual Information with the minimum of H(X) and H(Y)
* Able to detect both **linear** and **non-linear** dependencies (unlike Pearson)
* Applicable and very effective for use with **discrete features,** not continuous (will require adequate binning)

# Classification and Regression Techniques: Decision Tree Classifier and K-NearestNeighbours

Classification:

* Predicting the future based on historical data
* The foundation required for Machine Learning and Artificial Intelligence
* Given a collection of records or a training dataset, each record contains a set of **attributes** and **one class label**
* The goal is to find a **predictive model** for the class label as a function of other attributes

Decision Tree Basics:

* **Boolean tree structure**
* Each node represents **a test** on an attribute
* The branch represents **the outcome** of the test
* Each leaf node represents **class labels** or the **class distribution**

So how do we create a tree and where to split the records (**Decision Tree**)?

1. Multi-way split: Use as many partitions as distinct values
2. Binary Split: Divides values into two subsets (requires and optimal partitioning)

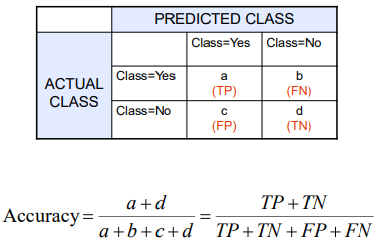
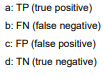
What about Continuous Attributes (**Decision Tree**)?

1. Discretization: Form an ordinal categorical attribute
2. Binary Decision: Split it under a condition (need to find the best fit)

How do we determine the **Best Split** then (**Entropy**) (**Decision Tree**)?

* Greedy approach - Nodes with **homogeneous** class distributions are preferred
* Split the data by level of impurity (unit of Entropy)

Metrics for Performance Evaluation (**Decision Tree**):



* For an accurate **decision tree classifier**, we want to minimise both False Positives and False Negatives
* Also need to consider that accuracy may be misleading if there is a significant discrepancy between sample size

K-NearestNeighbours:

* Essentially says “If it walks and quacks like a duck, then it’s probably a duck”
* Requires three conditions to work:

1. The set of **stored records**
2. **Distance Metric** to compute the distance between records
3. The value of *k* to retrieve

* To classify an unknown record:

1. **Compute the distance** to other training records
2. Identify *k* **nearest neighbours**
3. Use the **class label** of the nearest neighbour to determine the class label of the unknown record

* If *k* is too small, it becomes **sensitive to noise points**
* If *k* is too large, the neighbourhood may **include points from other classes**

# Data Linkage, Privacy and Bloom Filters

Data Linkage is the term coined for combining related/equivalent records across data sources (think of it as a join between two databases on a Primary Key)

Blocking:

* Prep:
  + Clean records / rows
* Block:
  + Represent complex records as simple values
  + Each record from A is allocated to n blocks and each record from B is allocated to m blocks
  + Within each block, compare records from A against those from B
  + If two records are not in the same block, it means they are **not a match**
* Score:
  + Score records with blocks in common
  + Calculate using Jaccard’s similarity (sum of common blocks / total number of blocks)
* Match:
  + Join records with that are sufficiently similar
  + Thresholds can be numerical values calculated from scores
* Merge:
  + Merge matching records
  + Resolve any conflicting attributes

Two Party Protocol:

* Each party applies a one way has function the attribute used to join the databases
* They then share the hashed value with the other party and check matching values. These are the linked records
* What happens about single character differences?
  + Single character differences usually result in a different hash function output
* The other party may also mount a **dictionary attack** to invert the hash function
  + By computing hash values for a range of values:
    - Party A can scan hash values received from Party B
    - Party A will then match with its given hash dictionary
    - If any match occurs, the privacy is breached

Three Party Protocol:

* To eliminate the possibility of a **dictionary attack**, we will involve a trusted third party
* Party A and B will both send their hashed values to Party C, who will check for matches for them
* But what if Party C is malicious?
  + Party A and B can apply **dictionary attack resistant hashing**
    - Party A and B will **agree** on a **salt for their data**
    - Party C should NOT know the salt and thus will not be able to perform a dictionary attack
  + Party A and B can also prevent **frequency attacks**
    - Party A and B can add random records / rows to manipulate the frequency distributions

So, what is the best way to keep privacy in check?

* Use a trusted third-party C to prevent a dictionary attack
* Use a one-way hash function with a salt to prevent a dictionary attack from the trusted third-party C
* Add random records to prevent a frequency attack from the trusted third-party C
* However, this only works for **exact matching** between attributes

Computing **approximate similarity** in a **privacy preserving manner**:

* If we want to compute the similarity between two strings from two different databases
  + Convert each string into n-grams (2-grams for the example)
  + Convert the string into lower case
  + Let h = number of common n-grams, x = number of n-grams in the first string, y = number of n-grams in the second string
  + Use the **dice coefficient** to calculate the similarity score
* **Bloom filters** is a better alternative to **similarity**:
  + Represented as an array initialised to 0
  + Use a hash function to map the first string into the array
  + We can then hash another string into the bloom filter
  + If any index of the hashed string has a 0, then it is definitely not part of the bloom filter and therefore not a match
  + If all indices of the hashed strings are set to 1, then it will appear to be a member of the bloom filter (but be aware of false positives since it may be a coincidence)
  + To calculate the **similarity of a bloom filter**, let h = number of indices set to 1 in the bloom filter, b1 = number of indices set to 1 in bloom filter B1, b2 = number of indices set to 1 in bloom filter B2
  + We still need to choose a threshold value for deciding whether to conclude that string A matches string B (or whether if string A is similar to string B)
  + **Privacy considerations**:
    - Party A and B agree on a fixed array length
    - Party A and B also agree on a hash function with salt
    - Party A and B may also add dummy records to reduce the chances of a successful frequency attack

# Blockchain and Data Processing

Blockchain is an infrastructure based on peer-to-peer, hashing and public key cryptography technology. It is a type of **distributed database** stored over **various computers** with **no central point of control**.

A **ledger** is:

* Publicly available
* Cannot be altered
* Can be verified for integrity
* Stores records, events, facts, asset transfers and etc

The main benefits of Blockchain is that there will be:

* Less administration, bureaucracy, etc
* Less expensive (fees or cost)
* Faster transactions (no middleman delay time)
* More control over records handed to users (you get to choose what to send)
* Ability to be anonymous
* Users can verify the integrity of data in the blockchain
* A more secure solution (maybe)

The **public ledger** is called **the blockchain** (a file), which is a **sequence of blocks** where each block contains a **header** and some **data** (a list of transactions or facts). The **block ID** is the hash of its reader and contains the ID of its parent block. Each computer in the **blockchain network** has a complete copy of the publicly available ledger.

The **block’s header** typically includes:

* ID of its parent block
* Timestamp of the block’s creation
* Hashed values of the data (list of transactions or facts) inside a block

Suppose we wish to add data to the blockchain:

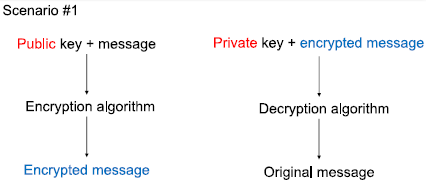
* Bob’s computer will broadcast the fact to the neighbours connected to in the blockchain network
* These neighbours receive the fact, verify/validate the correctness of its format and then broadcast it to its peers they are connected to. These peers then recursively follow the same procedure
* At some point, a peer will aggregate a collection of the facts (transactions) it has received and place them into a block, creating an appropriate header and then broadcast the block to its neighbours

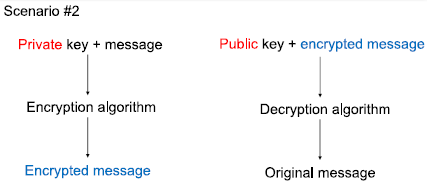
Chaining characteristics:

* The ID of a block is derived from the hash of its reader (so if a block’s header changes, the block’s ID also changes)
* If a parent block is modified, then its header will change resulting in a change of the parent’s ID
* If the parent ID changes, then this will change the previous block hash inside the child’s header
  + Since the child’s ID is derived from its own header, the child’s ID will change as a result
  + This in turn results in a change in the grandchild which is dependent on the child’s ID
  + Events keep continuing such that all children have a change in ID
  + **Changing a block produces a cascade effect requiring recalculation of all subsequent blocks**
* Nodes may create blocks simultaneously and propose they be added as the next block in the blockchain
  + We must resolve discrepancies to reach a consensus
* Nodes may try to disrupt the blockchain by creating a false fact (ie duplicate transactions despite only one being authorized)

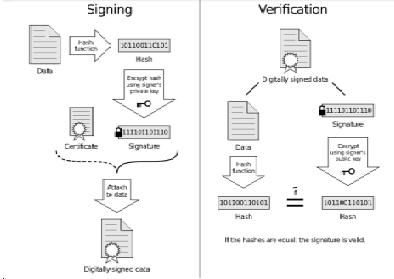
Public-Key cryptography:

* A private key (only known by the sender)
* A public key (known by the parties the sender wishes to share with)





But what if the data is sensitive and does not want it to be publicly viewable on the blockchain?



# k-Anonymity and Ɩ-Diversity

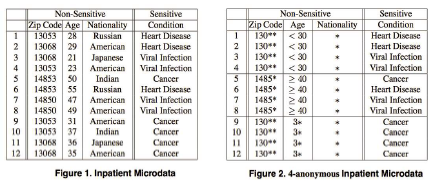
Terminology:

* **Explicit Identifier**: Unique identifier for an individual or PK
* **Quasi-Identifier**: A combination of non-sensitive attributes that can be linked with external data to identify an individual
* **Sensitive Attributes**: Information which people do not wish to reveal

k-Anonymity is used to produce a release of the data with scientific guarantees that the individuals who are the subjects of the data cannot be re-identified whilst the data remains practically useful.

A dataframe satisfies **k-anonymity** if every record in the table is indistinguishable from **at least k-1 other records** with respect to every set of quasi-identifying attributes. This is called a **k-anonymous table**.

For ever combination of values of the quasi-identifiers in the k-anonymous table, there must be at least *k* records that share the same values.



How do you achieve k-anonymity?

* Generalization:
  + Make the quasi-identifiers less specific or more broad
  + Example: Specific sports, musical instruments and academic subjects are summer into sports, music and studies
* Suppression:
  + Remove or supress the quasi-identifiers completely
  + Moderate the generalization process
  + Limited number of outliers
  + Can be used in row, column and cell levels.
  + Example: Post code 3104 and 3108 become 31\*\* and 31\*\*
* In the worst-case scenario when data gets into the wrong hands, at most, they will only be able to narrow it down to a quasi-identifier to a **group** of *k* individuals

**Attacks** on k-anonymity:

* Homogeneity Attack:
  + k-anonymity can create groups that leak information due to a lack of diversity in the sensitive attribute
  + Example: If person X knows that person Y is A years old from a B background and lives in town C, then person X can narrow down person Y to a quasi-identifier. If the sensitive information S is the same, person X can conclude regardless of anonymity that person Y has sensitive information S]
* Background Attack:
  + k-anonymity is unable to protect against attacks based on background knowledge
  + Example: If person X knows person Y comes from country C which has a low chance of disease A occurring, then person X can conclude that person Y is more likely to have disease B

The solution to the two main attacks is **Ɩ-diversity** which aims to make the sensitive attributes diverse within each group.

Location and Trajectory Privacy may also be breached via **Inference attacks** by monitoring person X’s travel route. We can learn about their habits, preference and location and narrow down the user if have additional information about them.

To avoid this, we can use **cloaking** which combines:

* **k-anonymity** which makes individuals k-anonymous if their location information cannot be distinguished from k-1 individuals
* **Spatial cloaking** which adapts the spatial precision of location information about a person according to the number of other people in the same quadrant or area
* **Temporal cloaking** which will reduce the frequency of temporal information available

This means that for a user query, they must wait for at least *k* number of other users requesting the same query until it is executed. This results in a very slow wait time.

Obfuscation combats the slow wait time by masking an individual’s precise location to an approximate degree. It will deliberately degrade the quality of information about an individual’s location. Works on the assumption that the more obfuscation radius, the more private the users location will be.



In summary, **k-anonymity** (individuals cannot be distinguished from k-1 other individuals) has a longer wait time but is much more accurate whilst **obfuscation** (the greater the imperfect knowledge, the greater the user’s privacy) is instant but more inaccurate.

To reduce the risk of re-identifying individuals in released datasets, we must choose a suitable value of *k* and manipulate it such that it becomes k-anonymous (through generalization or suppression) and Ɩ-diverse (ensure that there are at least *Ɩ* sensitive attributes in each group).

# Differential Privacy (Local and Global)

Global (Individual Data + Overall Noise -> Result)

* We have a sensitive dataset with a trusted owner person X and researcher person Y
* Person X will do the **analysis on the raw data** and then **add noise to the results**
* Person X will then report the noisy answers to person Y

Local (Individual Noise + Overall Data -> Result):

* Each individual is responsible for adding their own noise to the data
* Person X may take a survey but only choose to answer correctly given some condition

Whilst *k-anonymity*and *Ɩ-diversity*aim to take the original data and anonymize sensitive information, **differential privacy** aims to correctly analyse the original data, add noise and then release the correct results **with noise**.

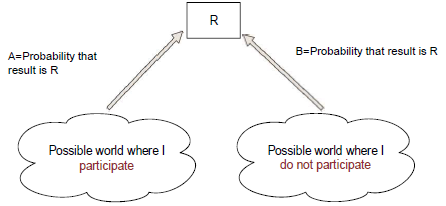
Privatised analysis:

* Query the data and **obtain the real result**
* Conduct an analysis on it
* Add **random noise** to hide the presence or absence of any individual
* Release the noisy data to the public



* Achieved by adding some random values but making sure that it is distributed with mean 0

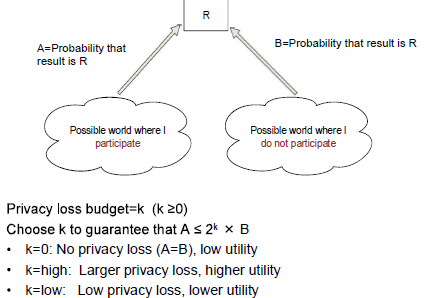
The **promise** of differential privacy is that the chance that the noisy released result will be result R is approximately the same, regardless if an individual participates or not. If you can guarantee that A≈B, then no one will be able to guess which possible world resulting in result R.



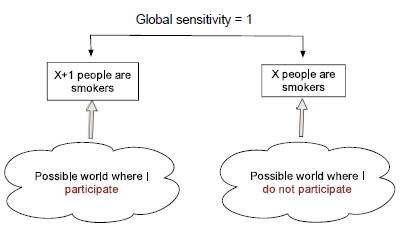
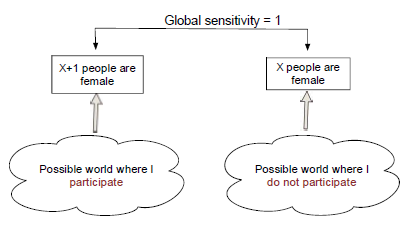
This means that an attacker will not be able to **learn anything sensitive** but may still be able to draw conclusions on a distribution of the data. Example: 120 females and 200 males took a subject X the attacker will be able to conclude the distribution of genders but not if a specific person A took the subject.

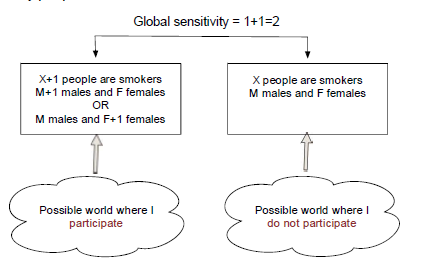
So **how much** noise should we add to the result?

* Privacy Loss Budget:
  + How private do you want the result to become?
  + How hard do you want to make it for the attacker to guess the true result?
  + Choose a value *k* such that the presence or absence of a user in a dataset will not have a **considerable effect** on the released result



* Global Sensitivity:
  + How much difference the presence or absence of an individual could make to the result
  + The global sensitivity of a query Q is the **maximum difference** in answers that is obtained when **adding or removing any individual from the dataset** (maximum effect of a single individual)
  + For multiple queries, the global sensitivity then becomes the sum of maximum differences

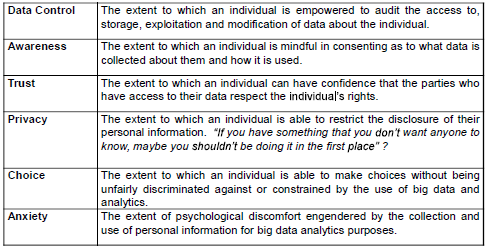


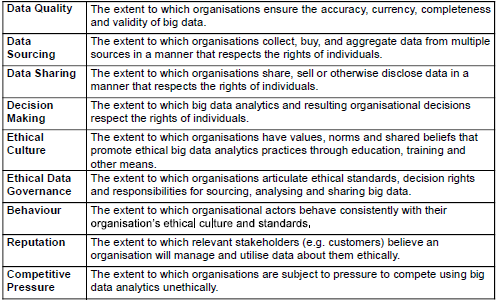


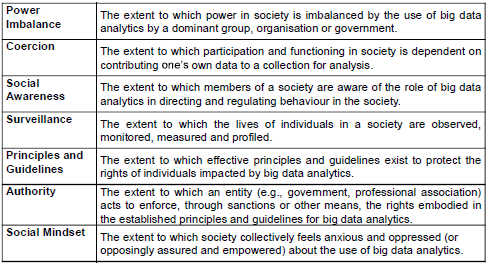
In summary, **differential privacy** guarantees that the **presence or absence of a user cannot be revealed** after releasing the results but does not prevent attackers from **drawing conclusions about individuals** from the aggregated results over the population.

# Big Data Analytics

BDA is the ability to collect, store and process increasingly large and complex datasets from a variety of sources, into competitive advantage.







Rules:

1. Acknowledge that data are people and can cause harm
   1. All data are people until proven otherwise
2. Recognise that privacy is more than just a quantifiable value
   1. Privacy is contextual and situational
   2. A single post on social media against the whole history of posts
   3. Privacy preferences differ across individuals and societies
3. Guard against the reidentification of your data
   1. Metadata associated with photos
   2. Reverse image searching
4. Practice ethical data sharing
   1. Seek consent from participants when sharing data or photos
5. Consider the strengths and limitations of your data (big is not always better)
   1. Document the provenance and evolution of your data
   2. Do not overstate clarity and acknowledge the challenges faced
6. Debate the tough, ethical choices or issues
   1. Importance of debating issues within groups of peers
   2. Examples include Facebook or Google
7. Develop a code of conduct for your organization, research community of industry
   1. Are we abiding the ToS of the user expectations?
   2. Does the general public consider what you do as creepy or privacy invasive?
8. Design your data and systems for auditability
   1. Plan for and welcome audits of your big data practices
   2. Systems of auditability clarify how different datasets differ and aid in making research better
9. Engage with the broader consequences of data and analysis practices
   1. Recognise that doing big data has a considerable societal effect
10. Know when to break these rules
    1. Emergency
    2. War
    3. Natural disasters