REVISION GUIDE

DECODED

OVERVIEW

This revision guide is divided into six (6) main topics reflected in the EMC exam. Within each exam section, there is material from the relevant modules and/or topics covered during the apprenticeship, including key definitions and practice questions.

SELF-ASSESSMENT GUIDE

Use this guide to assess your development level for each topic. This way you can focus on the areas where you are at a lower level of 1 or 2 at first before progressing to review the topics where you are at a higher development level.

PRACTICE QUESTIONS

Throughout this guide you'll find practice questions to help reinforce your learning. You can find model answers in a separate document.

BIG DATA, ANALYTICS & DATA SCIENTIST ROLE

1. BIG DATA

OVERVIEW

This section summarises the characteristics and business drivers of Big Data

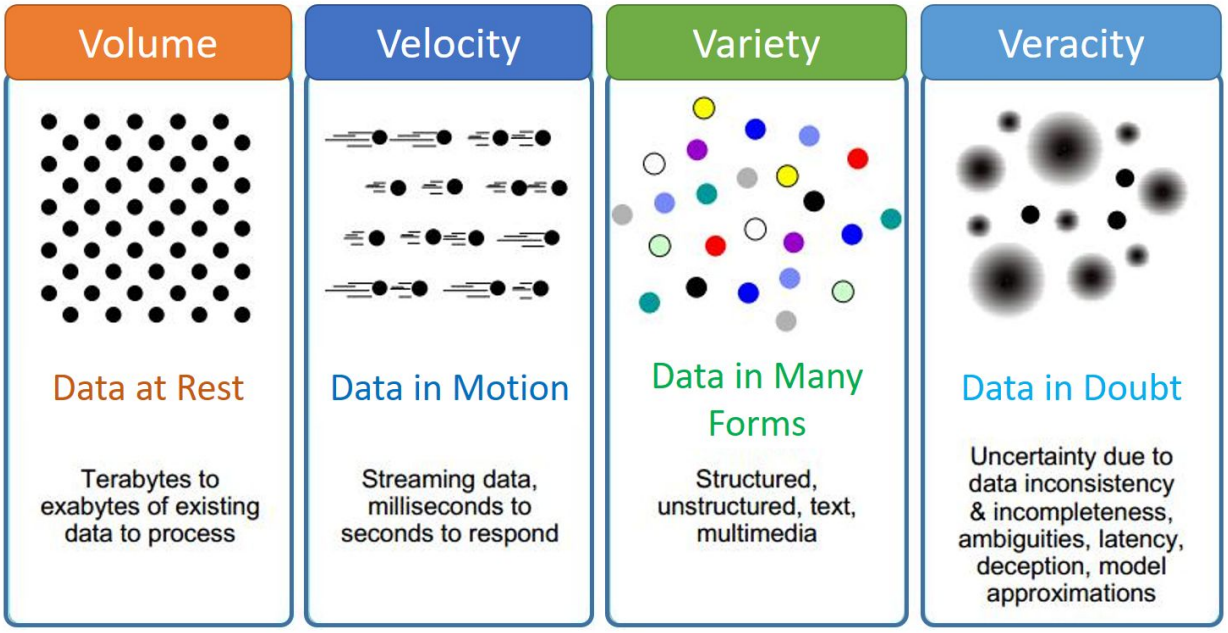
KEY CONCEPTS & TERMINOLOGY

Characteristics of Big Data: The 4 V's

Sources of Big Data

Data Ecosystem

CHARACTERISTICS OF BIG DATA: THE "4 V'S"



VOLUME

The main characteristic that makes data "big" is the volume. The total amount of information being produced and collected is growing exponentially every year.

VELOCITY

Velocity is the frequency of incoming data that needs to be processed. With lots of data comes the need to be able to aggregate and disperse data extremely fast.

VARIETY

The variety of data types in Big Data ranges from the traditional structured types (e.g., bank statements, spreadsheets) to more untraditional, relatively newer unstructured types (e.g., text,

images, twitter feeds, audio files, web pages). Unstructured data is considered a core concept of Big Data.

VERACITY

This is the “trustworthiness” of the data. In other words, is the data reliable enough to use in your business. While there are always discrepancies in the data, the concept of veracity is to ensure the data is as reliable as possible.

OPTIONAL 5TH: VALUE

The value of data underlies the veracity, variety, volume, and velocity of Big Data. It is always worth considering if the data being used is valuable to any business-related aims or goals.

SOURCES OF BIG DATA

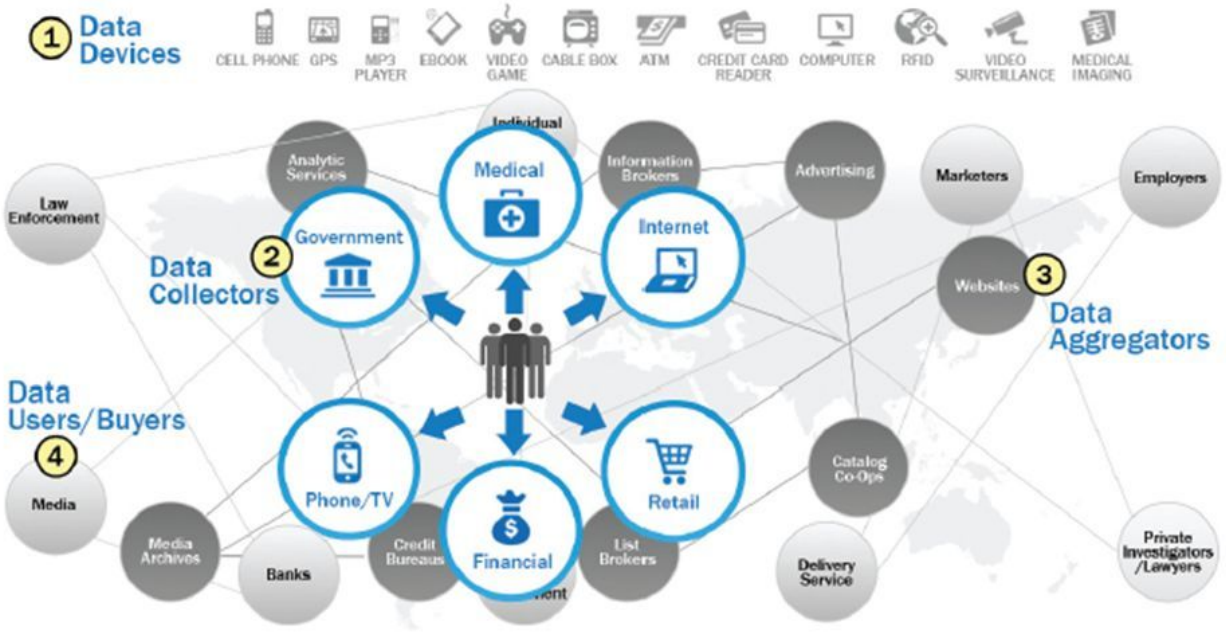
Big Data is driven by many sources of information, including:

- Mobile sensors/devices
- Social media
- Video surveillance
- Video rendering
- Smart grids/buildings
- Geophysical exploration
- Medical imaging
- Genetic sequencing
- Internet photo & video footage
- Nontraditional IT devices (e.g., GPS)

Social media and genetic sequencing are some of the fastest-growing sources of Big Data in modern times. Because of the vast amount of information being collected on a daily basis, the **data ecosystem** has changed to accommodate these rapid growths.

DATA ECOSYSTEM

The data ecosystem is a collection of infrastructures, analytics, and applications that are used to collect, organise, and analyse a vast amount of data. The ecosystem is split up into four groups: **data devices**, **data collectors**, **data aggregators**, and **data users & buyers** (see diagram below):



1. **Data Devices:** Data devices are constantly gathering information. These can range from computer and gaming devices, to medical records and store loyalty cards. Anything that can be a source of information can be considered a data device.

2. **Data Collectors:** These are items that collect any sort of data from devices (see point 1 above) and users. These can include retail/online stores; financial institutions (e.g., banks, insurance companies); phone and TV providers; web browser history; medical facilities (e.g., hospitals); and government agencies (e.g., census).

3. **Data Aggregators:** Data aggregators help make sense of the data collected from the different entities (see point 2). These companies compile data from devices and collectors (see points 2 and 3 above). They can then transform and package the information to sell to others who can use the information (e.g., selling to list brokers so they can determine who are good targets for specific ads and campaigns).

4. **Data Users & Buyers:** The users and buyers are those who directly benefit from the data collected and aggregated by others (see points 2 and 3 above). Example buyers and users include media companies and social media websites, retail banks, credit bureaus, government bodies (e.g., presidential campaigns), and list brokers.

PRACTICE QUESTIONS:

1. Describe the main characteristics of Big Data.

2. What are the 4 parts of a Data Ecosystem, and what is unique to each part?

2. INTRODUCTION TO DATA

OVERVIEW

This section summarises the different kinds of data structures, the data scientist role (compared to business intelligence analysts), and the business drivers of data

KEY CONCEPTS & TERMINOLOGY

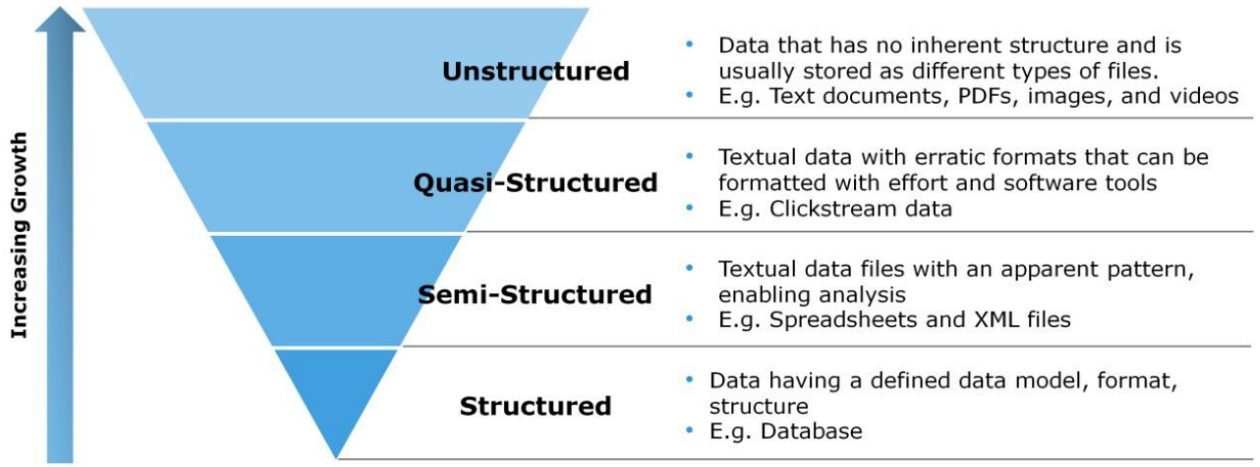
Data Types

Data Scientist Role

Business drivers of data

DATA TYPES

There are four main types of data: structured, semi-structured, quasi-structured, and unstructured. Much of the data we use falls into the structured or unstructured types, but it is good to be aware of all four.



STRUCTURED DATA is well defined, and has a set of rules guiding its set-up. A prime example of structured data is a *database*. Other examples include spreadsheets and *dates* (which always follow a specific pattern).

QUASI-STRUCTURED DATA is textual data with irregular formats, but can be formatted with the appropriate tools. Examples of quasi-structured data includes information about a user's web-page history *(webserver logs)* and clickstream data.

SEMI-STRUCTURED DATA is a *form of structured data* that does not obey a normal, formal structure generally associated with databases or data tables. It includes textual files (like XML files) that have some recognisable pattern that allows for parsing of information.

UNSTRUCTURED DATA has no set "rules" or coherent structure, and is not organised in any particular format. Examples include *pictures, PDF files, voice recordings, tweets, and websites*. One of the goals of Big Data is to take unstructured data and give it "structure" so a computer can understand it for further analysis and interpretation.

DATA SCIENTIST ROLE

While the roles of data science and business intelligence (BI) might seem similar, there are some distinct responsibilities that differ between the two roles. They differ in both their analytical approach, the types of data sources they use, and the time frame on which their enquiries are focus on (i.e., prospective/future vs. retrospective/past).

Data Science	***Business Intelligence***
Provides insight & foresight: Optimisation, predictive modelling, forecasting, statistical analysis	*Provides hindsight and insight:* Standard & *ad hoc* reporting, dashboards, alerts, queries, details on demand
Variety of *structured & unstructured* data	Manageable, *highly structured* data
Focuses on the *present/future* & *how/why* ● "What if…?" ● "What is the optimal scenario for the business?" ● "What will happen next?" ● "What if these trends continue?" ● Why are these trends happening?"	Focuses on the *past/present* & *when/where* ● "What happened last quarter?" ● "How many units were sold last month/quarter/year?" ● "Where is there a business problem?" ● "In which situations is a problem occurring?"

A Data Scientist should have five main skills/characteristics:

1. Quantitative skills in mathematics and/or statistics
2. Technical aptitude for programming, machine learning, and/or software engineering
3. Skeptical mind-set & critical thinking skills to examine your own work
4. Curious and creative problem-solving skills
5. Can communicate (and collaborate) business values clearly with various stakeholders

BUSINESS DRIVERS OF DATA

Many business problems can be solved with the use of data-driven, analytical techniques. Below are four common business problems where the use of advanced analytics can create competitive advantages by deriving more valuable and actionable insights.

Business Drivers	***Examples***
Optimise business operations	Sales, pricing, profitability, efficiency
Identify business risks	Customer churn, fraud, default
Predict new business opportunities	Upsell, cross-sell, new customer prospects
Law/regulatory requirements compliance	Anti-money laundering, fair lending, SOX

PRACTICE QUESTIONS:

1. What are the four types of data structures? Provide an example for each type.

2. Describe three differences between data science and business intelligence roles.

DATA ANALYTICS LIFECYCLE

INTRODUCTION TO DATA

OVERVIEW

This section summarises the data analytics lifecycle and explores the different techniques in data mining.

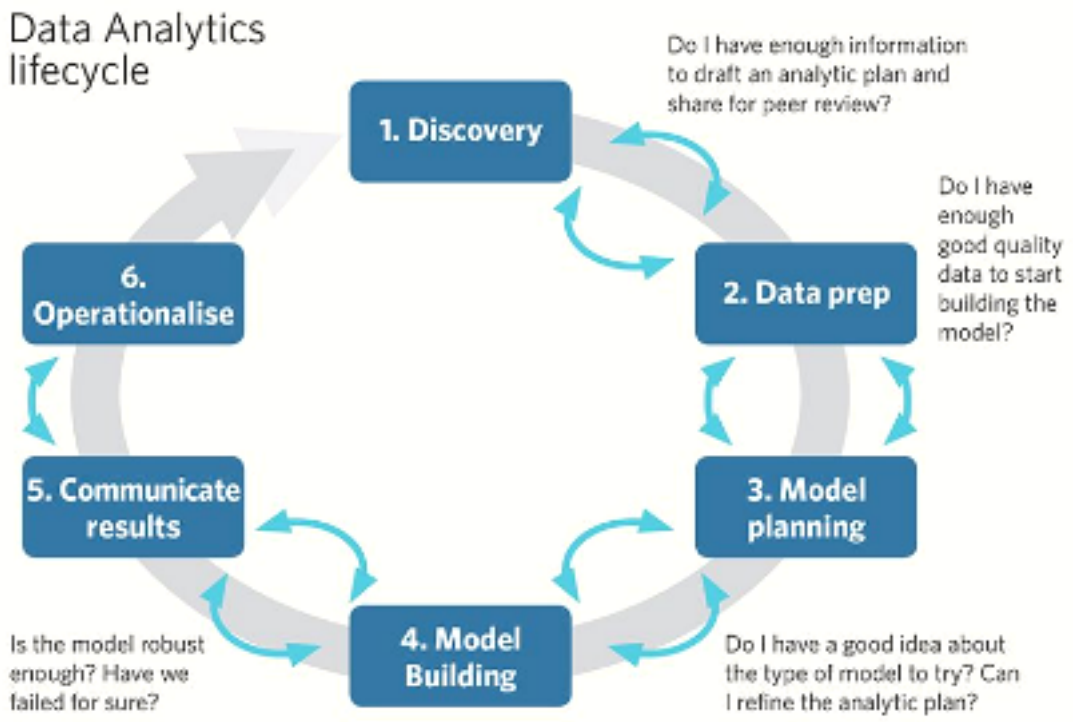
KEY CONCEPTS & TERMINOLOGY

Overview of the Data Analytics Lifecycle

6 Phases of the Data Analytics Lifecycle

OVERVIEW OF THE DATA ANALYTICS LIFECYCLE

The Data Analytics Lifecycle is designed specifically for Big Data problems and data science projects. The lifecycle has six phases, and project work can occur in several phases at once. For most phases in the lifecycle, the movement is not perfectly linear - there is a constant flow between phases, and progress can move either forward or backward at any given time.



PHASE 1: DISCOVERY

In Phase 1, the team learns the business domain, including relevant history such as whether the organisation or business unit has attempted similar projects in the past from which they can learn. The team assesses the resources available to support the project in terms of people, technology, time, and data. Important activities in this phase include *framing the business problem* as an analytics challenge that can be addressed in subsequent phases, and *formulating initial hypotheses* to test and begin learning from the data.

PHASE 2: DATA PREPARATION

Phase 2 *requires* the presence of an *analytic sandbox**, in which the team can work with data and perform analytics for the duration of the project. The team needs to *execute **ELT***: “extract, load, and transform” (or extract, transform, and load [**ETL**]). Data should be transformed in the **ELT/ETL** process so the team can work with it and analyse it. In this phase, the team also needs to *familiarise itself with the data thoroughly* and take steps to *condition the data*.

**A data sandbox (in big data) is a scalable and developmental platform used to explore an organization's many data sets through interaction and collaboration. It allows a company to realize its actual investment value in big data. This needs to be completed before starting phase 2.*

PHASE 3: MODEL PLANNING

Phase 3 is model planning, where the team *determines the methods, techniques, and workflow* it intends to follow for the subsequent model building phase. The team explores the data to learn about the *relationships between all the variables*, and subsequently *selects key variables* and the *most suitable models*.

PHASE 4: MODEL BUILDING

In Phase 4, the team *develops datasets for testing, training, and production purposes*. In addition, the team builds and *executes models* based on the work done in the model planning phase. The team also considers whether its existing tools will suffice for running the models, or if it will need a more robust environment for executing models and workflows (e.g., fast hardware and parallel processing, if applicable).

PHASE 5: COMMUNICATE RESULTS

In Phase 5, the team - in collaboration with major stakeholders - *determines if the results* of the project *are a success or a failure* based on the criteria developed in Phase 1. The team should *identify key findings, quantify the business value,* and *develop a narrative* to summarise and convey findings to stakeholders.

PHASE 6: OPERATIONALISE

In Phase 6, the team *delivers final reports, briefings, code, and technical documents*. In addition, the team *may run a pilot project* to implement the models in a production environment.

PRACTICE QUESTIONS:

1. Describe the 6 phases of the data lifecycle.

2. What is a data sandbox? In what phase is it required?

RECOMMENDED READING:

In case you want to explore more topics about data analytics lifecycle we recommend the following links:

- Big Data Analytics - Data Life Cycle
- An Approach to Machine Learning and Data Analytics Lifecycle
- The 6 Stages of Data Processing Cycle

INITIAL ANALYSIS OF THE DATA

INTRODUCTION TO STATISTICS

OVERVIEW

This is a one-page summary about probability, z-tests, Welch's t-test, Wilcoxon Rank Sum tests, and Analysis of Variance.

KEY CONCEPTS & TERMINOLOGY

Sample & Population Distributions	Z-tests
Student's t-test	Welch's t-test
Mann-Whitney U test	ANOVA tests
Statistical Equations	Non-parametric data

SAMPLE & POPULATION DISTRIBUTIONS

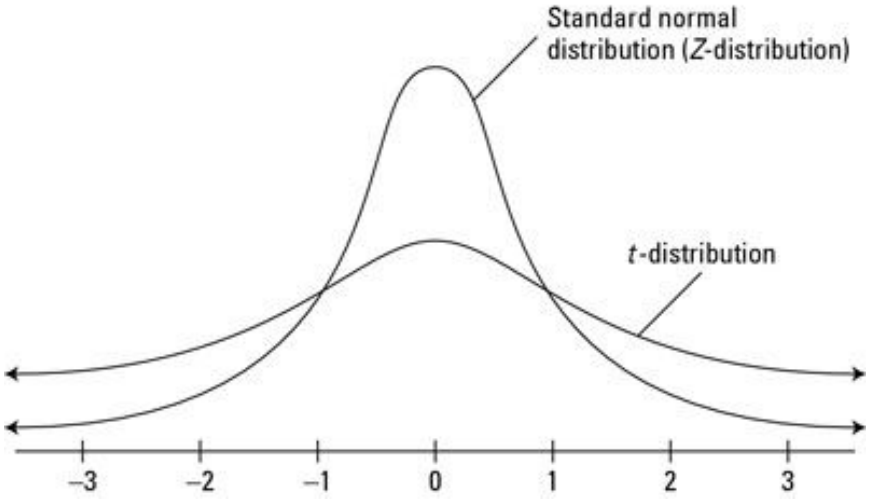
The z-distribution represents a total 'population' with known information, while the t-distribution represents a sample from a known or unknown population. A random sample (t-distribution) that comes from a larger population (z-distribution), is considered to be representative of that population.

These samples are considered ***parametric*** as they have normally distributed data, as seen in the figure below.

Z-distributions have normal means and standard deviations, and are from a known population

As sample sizes for t-distributions get larger, their parameters more closely resemble z-distribution parameters

Z-tests perform in a similar way as t-tests, but is rarely ever used in real-world contexts.



Z-TEST

Tests whether the mean of a *random sample* is the same (H_0), or different (H_A) from the population mean.

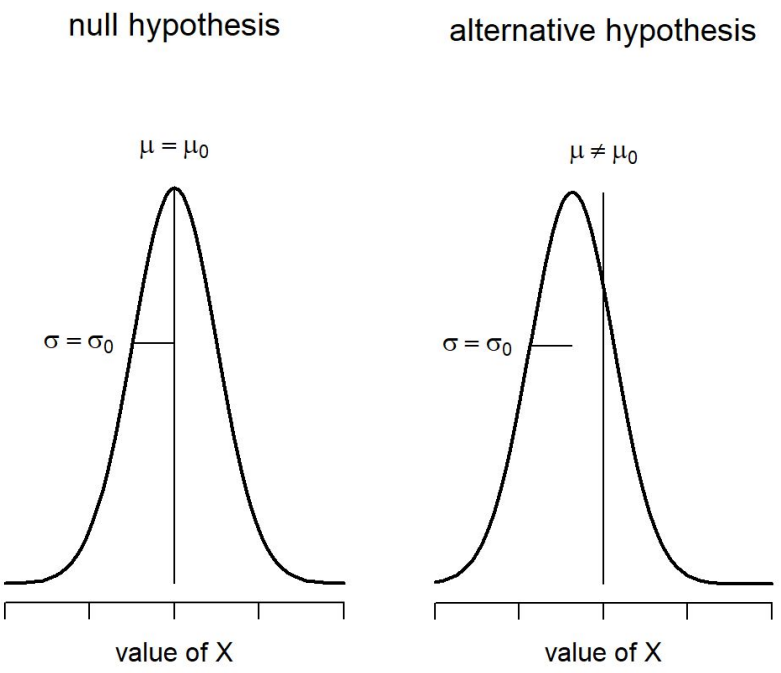
Null hypothesis (H_0):
The sample mean ($\boldsymbol{\mu_0}$) and is equal to the population mean ($\boldsymbol{\mu}$).

This means the random sample is actually from the population.

Alternative hypothesis (H_A):
The sample mean ($\boldsymbol{\mu_0}$) is *not equal* to the population mean ($\boldsymbol{\mu}$).

This means the random sample is *not* from the population.

Assumptions: The random sample and population are normally distributed, have known standard deviations ($\boldsymbol{\sigma}$), and samples are independent.



STUDENT'S T-TEST

Tests whether *two independent* samples have equal means (H_0) or unequal means (H_A). In various t-tests (including Student's t-test), the sample mean '$\boldsymbol{\mu}$' is also represented as '$\bar{x}$'. The z-test is similar to a one-sample t-test, except the $\boldsymbol{\sigma}$ values are unknown.

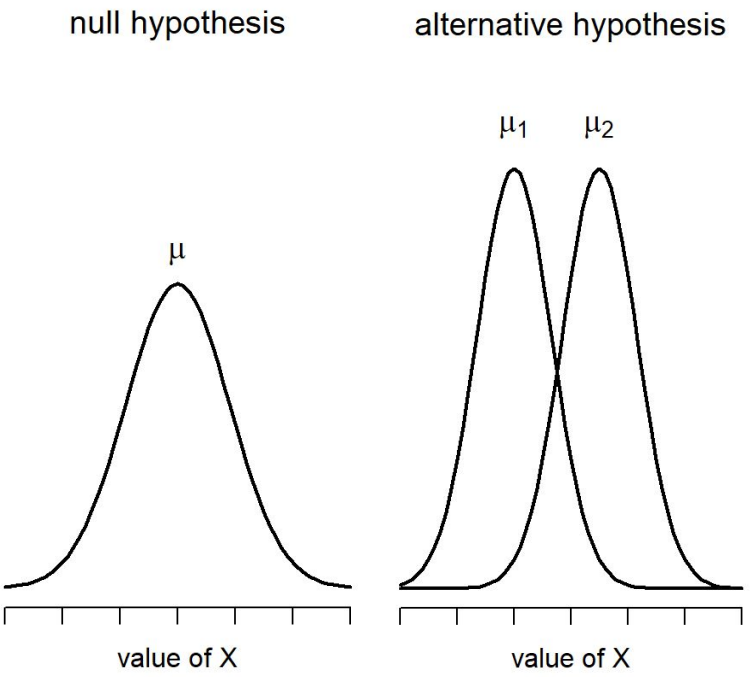
Null hypothesis (H_0):
The sample means for two independent samples ($\boldsymbol{\mu_1}$ and $\boldsymbol{\mu_2}$) are *equal* to each other.

This means the two samples are the same, and thus come from the same population.

Alternative hypothesis (H_A):
The mean of one sample ($\boldsymbol{\mu_1}$) is *not equal* to the mean of a second sample ($\boldsymbol{\mu_2}$).

This means the two samples are different from each other, and thus come from different populations.

Assumptions: The two samples are normally distributed, have the same standard deviations, and are independent of each other.



WELCH'S T-TEST

Tests whether *two independent samples* have equal means (H_0) or unequal means (H_A). It is an adaptation of the Student's t-test. They are more reliable when *samples have unequal variances* and/or unequal sample sizes. It also gives similar findings as a student's t-test when samples have equal variances.

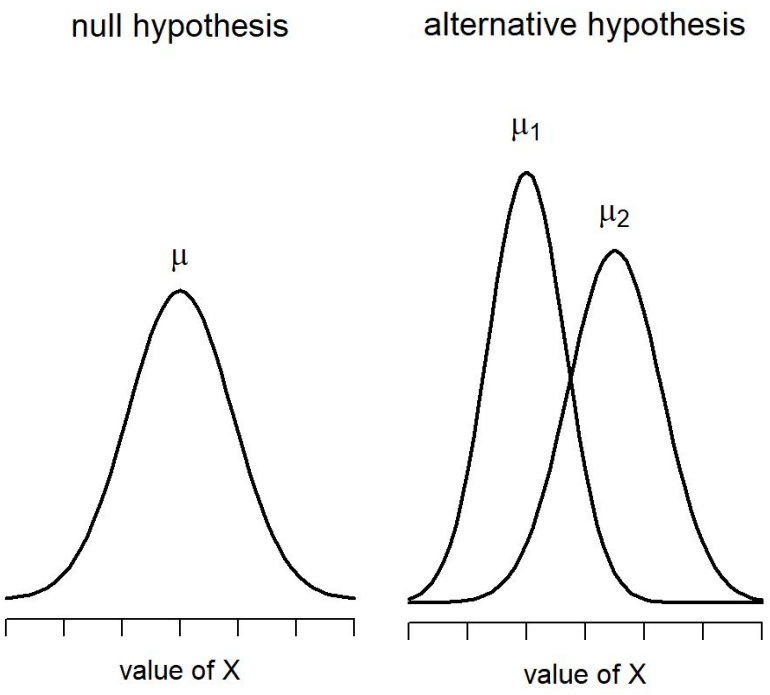
Null hypothesis (H_0):
The sample means for two independent samples ($\boldsymbol{\mu}_1$ and $\boldsymbol{\mu}_2$) are *equal* to each other.

This means the two samples are the same, and thus come from the same population.

Alternative hypothesis (H_A):
The mean of one sample ($\boldsymbol{\mu}_1$) is *not equal* to the mean of a second sample ($\boldsymbol{\mu}_2$).

This means the two samples are different from each other, and thus come from different populations.

Assumptions: Two samples are normally distributed and are independent. Standard deviations for the two samples are different (i.e., *unequal variances*).



MANN-WHITNEY U TEST (WILCOXON RANK SUM TEST)

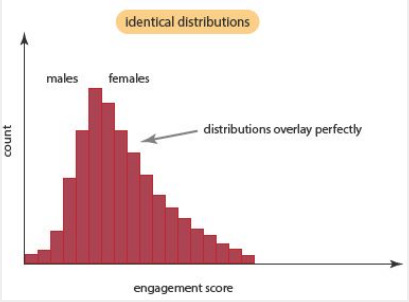
Test to determine whether *two independent samples* selected from two different populations (e.g., 'A' and 'B') have the same (H_0) or different (H_A) *distributions* (e.g., mean ranks).

Used when:

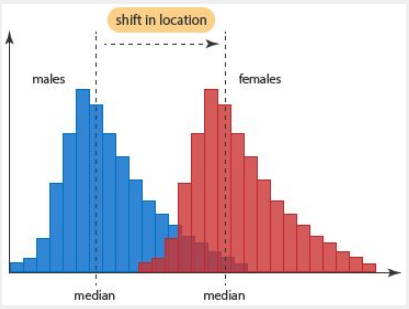
- Data does not fulfill normality assumptions (i.e., data is ***non-parametric***)
- Small sample sizes

The distribution of the samples are either 'free' (non-normal) or unspecified (unknown). However, it is less powerful than the standard t-tests for this reason.

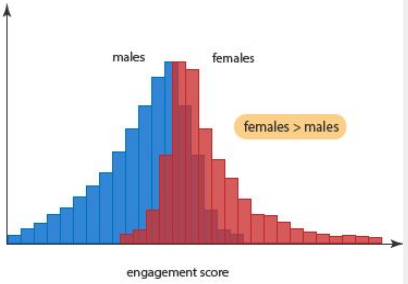
Null hypothesis:



Alternative hypothesis (ex. 1):



Alternative hypothesis (ex. 2):



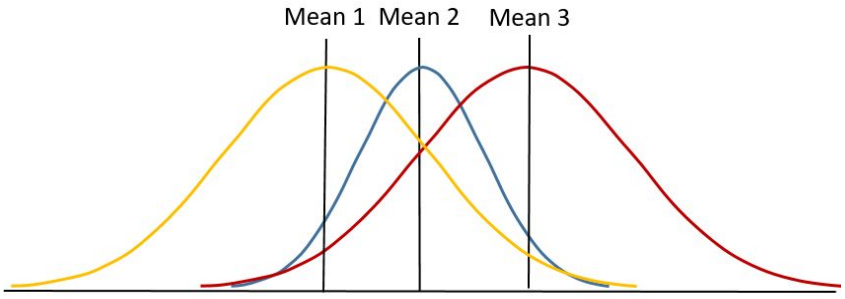
ANALYSIS OF VARIANCE (ANOVA) TESTS

Used to analyse if there are means differences between three or more groups

Types of ANOVAs		
One-way 1 independent variable affecting 1 dependent variable	*Two-way* 2 independent variables affecting 1 dependent variable	*MANOVA (multivariate ANOVA)* 1 or 2 independent variables affecting 3 or more dependent variables
Independent variable: the different groups/levels (i.e., categories) of a categorical variable		
Dependent variable: a continuous variable, where the mean of this value might differ by group		

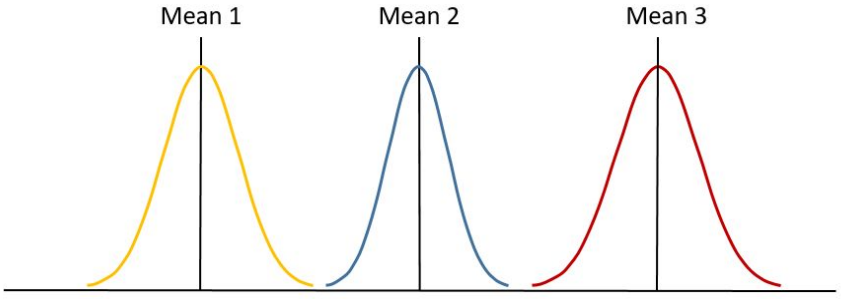
Null hypothesis (H_0):
The group means are similar or equal across all the groups (*top panel*)

This means the three (or more) samples are the same, and thus come from the same population.



Alternative hypothesis (H_A):
At least one group mean differs from the other group means (*bottom panel*)

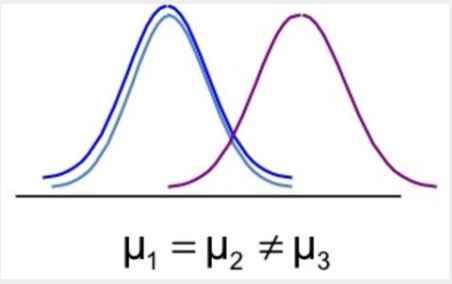
This means that at least one of the three samples is different from the rest, and thus come from different populations.



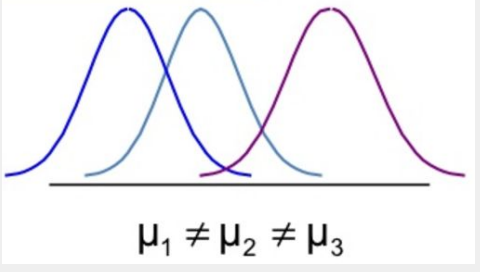
Assumptions: the normal distribution assumptions are met for all groups; all groups must have the same or similar sample sizes

Other examples of ANOVA alternative hypotheses (see images, right):

Alternative hypothesis (ex. 1):



Alternative hypothesis (ex. 2):



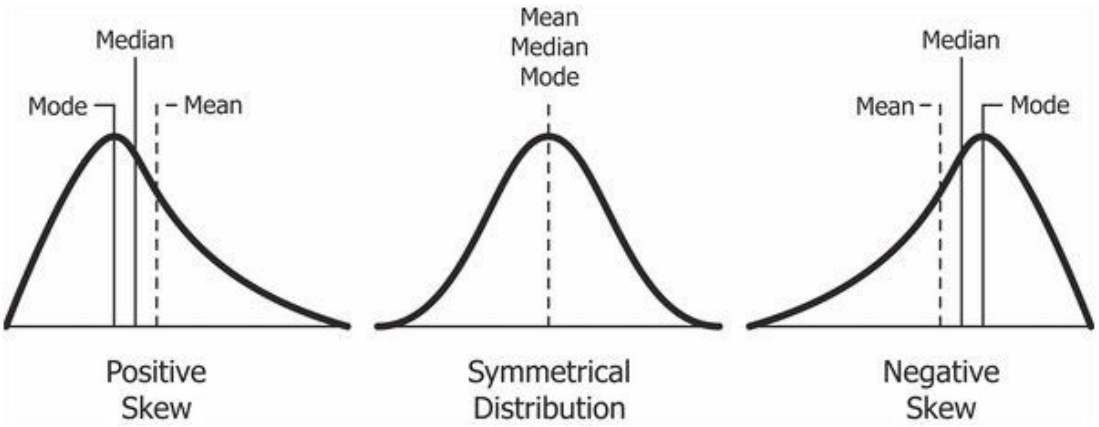
NON-PARAMETRIC DATA

It was briefly mentioned that *non-parametric* data is used in Mann-Whitney U tests. By non-parametric, we mean the data is not a normally-shaped, symmetric bell-curve, as seen in many of the example images discussing different t-tests. There are two measures that can help determine whether the data is parametric (normally distributed) or non-parametric (not normally distributed). These are **skewness** and **kurtosis**, which measure how different your distribution is from a theoretical, normal distribution.

SKEWNESS

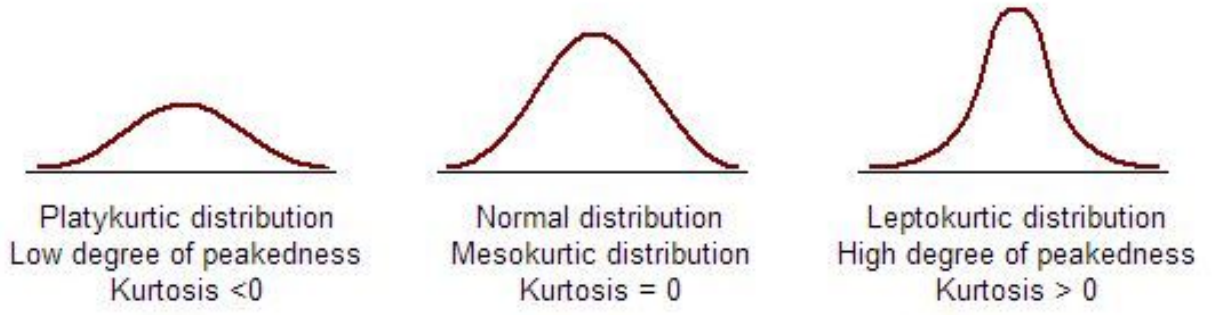
The *skew* of a distribution refers to the symmetry of the distribution. If you were to fold your bell-curve in half along a central line, would they be symmetric? If they are, there is no skew, so the value of skewness is zero. However, skewed data looks non-symmetric; there is likely to be long "tail" of data on one side or the other of the bell curve. The more non-symmetric it is, the higher the absolute value of skewness. Skewness can be either have a *positive value* (tail on the right side of the curve) or *negative value* (tail on the left side of the curve).

Additionally, in a parametric curve, the mean, median, and mode are the same value. In skewed data, the mean, median, and mode are all different values. See the image below for examples:



KURTOSIS

Kurtosis measures how tall or flat the bell curve is compared to the theoretical normal distribution. In a normal distribution bell curve, kurtosis has a value of zero. A kurtosis value larger than zero means the center peak of the curve is taller and thinner compared to the normal distribution - this is called *leptokurtic*. A kurtosis value smaller than zero (i.e., negative values) means the center peak of the bell curve is shallower and wider compared to a normal distribution - this is called *platykurtic*. See the images below for examples of kurtosis:



STATISTICAL EQUATIONS

Below are a few basic statistical equations you should be able to recognise:

Term	*Equation*
Population mean	$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$
Population standard deviation	$\sigma_N = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2}$
Population variance (standard deviation squared)	$\sigma_N^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2$
Sample mean	$\bar{x} = \frac{\sum x}{n}$ *or* $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$
Sample standard deviation	$\sqrt{\frac{\sum(x-\bar{x})^2}{n-1}}$ *or* $\sigma = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2}$
Sample variance (standard deviation squared)	$\sigma^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$

N = population size; *n* = sample size

PRACTICE QUESTIONS:

1. How do the following differ from each other:
    a. z-test and Student's t-test?
    b. z-distributions and t-distributions?

2. How does the Wilcoxon test differ from:
    a. Student's t-test?
    b. Welch's t-test?

3. Aside from ANOVAs, what other statistical tests require normal distribution assumptions be met?

4. In what situations would ANOVAs be more useful than t-tests or Wilcoxon tests? (Hint: think sample sizes, numbers of groups, etc.)

ADVANCED ANALYTICS: THEORY & METHODS

1. CLUSTERING

OVERVIEW

This summarises clustering methods, and specifically, the *k*-means algorithm.

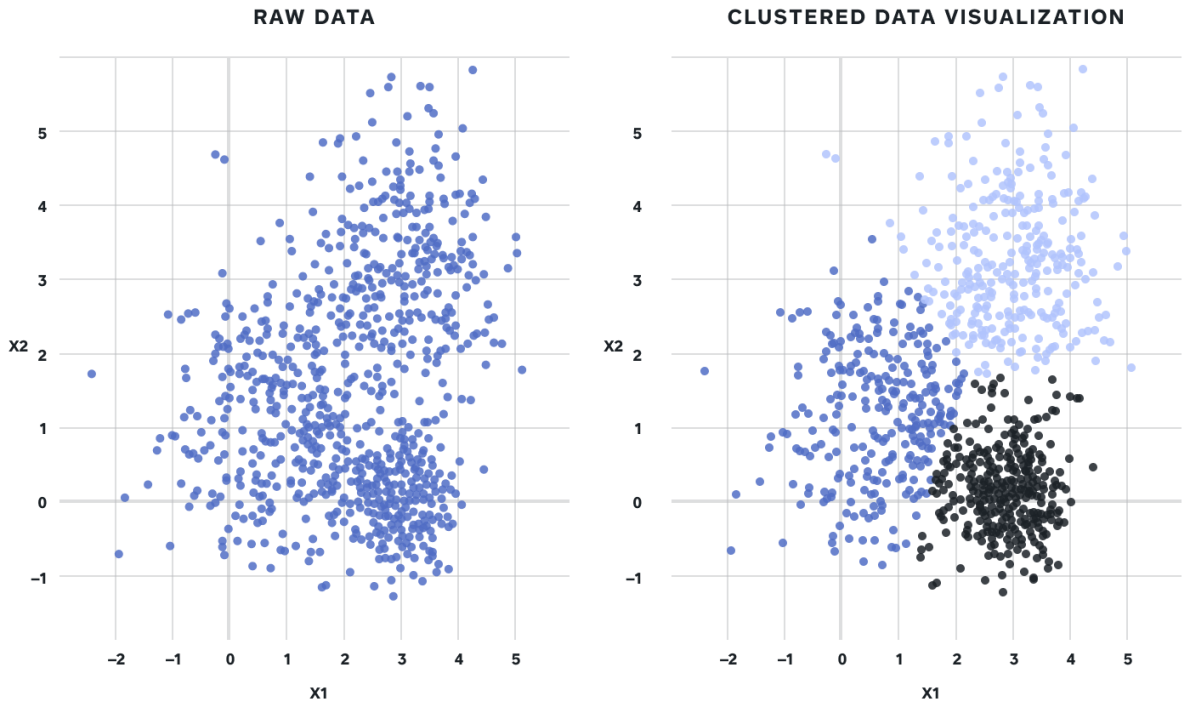
KEY CONCEPTS & TERMINOLOGY

Clustering	*k*-Means
Centroid	Within Sum of Squares (WSS)
Elbow Curve	Clustering in Multi-dimensions

CLUSTERING

Clustering is a method of exploratory data analysis that identifies groups of similar data points. Clustering is used to find hidden patterns in *unlabelled* data (that we as humans cannot easily identify), and thus groups similar data points together, in different unique "*clusters*", or groups.

In the image below, the raw data (before clustering) is on the left, and clustered data (after clustering) is on the right. This image shows how the data is split into 3 distinct clusters, each with unique characteristics that make it different from other clusters.



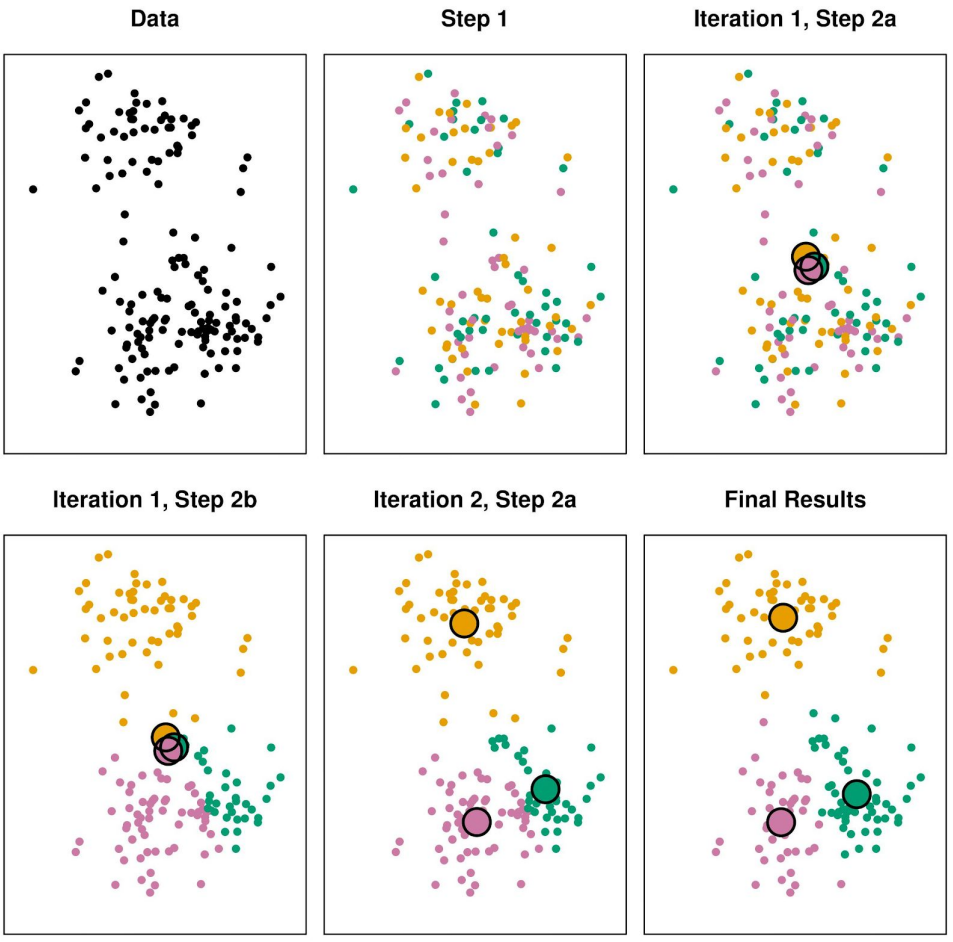
There are a number of methods that can be used to perform clustering analysis. Here, we will focus on one method called *k*-means.

K-MEANS ALGORITHM

The *k*-means algorithm is a technique that identifies a number clusters, based on objects' proximities to the center points (centroids) of '*k*' number of clusters.

CENTROID

k-Means works by minimising the distance between every data point (small coloured points in image below) and a hypothetical ***centroid*** (mean of all points belonging to a cluster; large coloured circles).

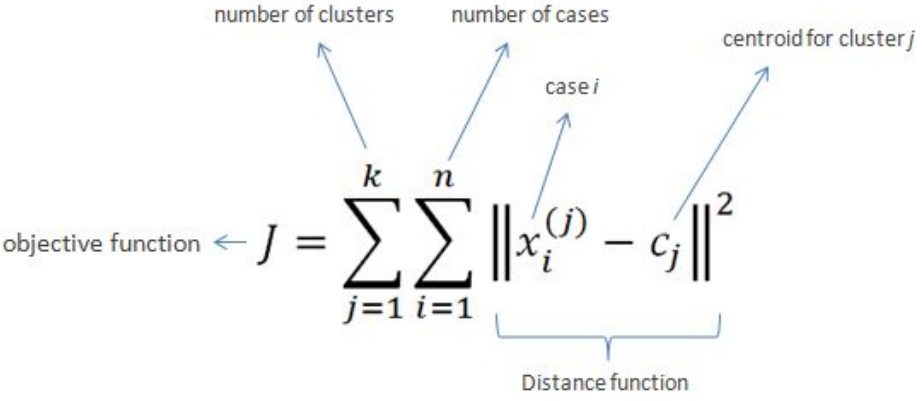


For a given value of *k*, the calculations to choose the right centroid position is done repeatedly until the optimal positions for the centroids have been identified, and every data point is assigned to a cluster. See this link for the step-by-step process.

The further an object is from a centroid, the less likely it is to be part of that cluster. The closer an object is to a centroid, the more likely it is to be part of that cluster.

WITHIN SUM OF SQUARES (WSS)

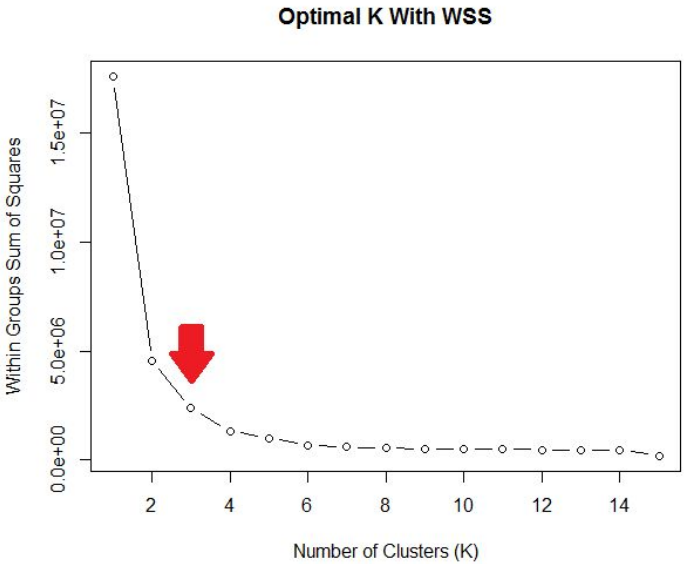
While the number of clusters is a value the data scientist will choose, there are methods to determine whether a certain number of clusters is optimal. One method is to use the Within Sum of Squares (WSS) metric, which helps determine the optimal '*k*' number of clusters.

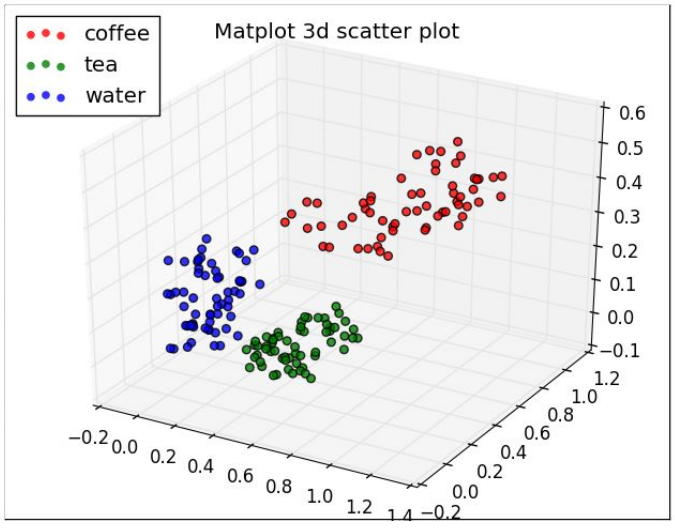


WSS is the sum of squares of the distances between each data point and the closest centroid. If data points are all close to their centroid, WSS is small; if data points are far from their centroid, the WSS is large. Ideally, selecting the optimal number of clusters reduces WSS. For instance, if having *k*-clusters of 3 results in a small WSS, and increasing *k* does not reduce WSS any more, then 3 would be the optimal value of *k*.

ELBOW CURVE

The elbow curve (right image) allows us to visualise the optimal *k* number of clusters for our data. It is a graph that plots any number of clusters on the x-axis you want to test, and the corresponding WSS values (based on these *k*-values) on the y-axis. Ideally, the optimal number of clusters will be there the WSS has the greatest reduction, but before the point where increasing the number of clusters does not significantly lower WSS any more.





CLUSTERING IN MULTI-DIMENSIONS

The same clustering principle can be applied to data in more than 2 dimensions (i.e., beyond a typical x-axis and y-axis graph). A 3-dimensional graph - which has an added z-axis - is just one more application of k-means clustering to a dataset with many features (left image). While the process is the same, the centroid is the mathematical mean of the data from all three dimensions (x-, y-, and z-axis) rather than just the x- and y-axis alone (as in 2-dimensional graphs).

PRACTICE QUESTIONS:

1. What does the '*k*' in *k*-means represent?
2. What is a centroid and why is it important in *k*-means?
3. How do you determine the optimal/best value for *k*?

2. ASSOCIATION RULES

OVERVIEW

Association rule mining is an unsupervised learning technique used to find hidden relationships in a dataset. The relationships derived are rule-based, meaning that they come in the form of *"If event A occurs then so will event B".* A real-life example of this would be: *"If customers buy milk, they will also buy butter.*" Additional information can be found at this website.

KEY CONCEPTS & TERMINOLOGY

Sparse Matrix	Transactions
Itemsets	Apriori Algorithm
Support	Lift
Confidence	Leverage
Conviction	

SPARSE MATRIX

A sparse matrix is a binary matrix (i.e. table) of 'true' (or 1) and 'false' (or 0) values. The defining characteristic of a sparse matrix is that there are more 'false' (or 0) values than 'true' (or 1) values (hence a 'sparsity' of information in the table) - this is in contrast to a dense matrix, where the majority of values are 'true' (or 1).

Sparse matrices are useful because they can be efficiently stored using compressed data storage formats, in which the computer only has to know where the 'true' values are located in the rows and columns - this greatly reduces storage costs and increases processing speed. This is necessary with association rule mining, which is often performed with huge datasets!

TRANSACTIONS

A transaction is a row of the "items" present in one data point (e.g., person 1 had one transaction with milk and cereal). In a retail context, the transactions might be the customer receipt of the items bought.

Below is an example transaction dataset:

	Milk	*Cereal*	*Cheese*	*Bread*
Person 1	1	1	0	0
Person 2	1	1	0	1
Person 3	0	0	1	1
Person 4	1	0	0	0
Person 5	1	0	1	1

ITEMSETS

An itemset is the set of items in a given transaction. An itemset can consist of a single item, double items, or multiple items. While this can go up to triple, quadruple, or even quintuple itemsets, we tend to begin with smaller itemsets and work our way up as we will discuss below.

Example itemsets		
Single itemset	*Double itemset*	*Triple itemset*
Person 4: {milk}	Person 1: {milk, cereal} Person 3: {cheese, bread}	Person 2: {milk, cereal, bread} Person 5: {milk, cheese, bread}

Each unique itemset can only occur once for any transaction. In our example, milk can only exist once in a given transaction, as the frequency (number) of an item is not accounted for in the sparse matrix - only whether or not the item was present is important.

A note on itemset notation:

An association rule between A and B is often denoted as ***{A} ⇒ {B}.***

Note that 'A' or 'B' could be any type of itemset - single, double, triple, etc.

Ex. 1: 'People who buy milk also buy cereal' is denoted by: *{milk} ⇒ {cereal}*

Ex. 2: 'People who buy bread & cheese also buy milk' is denoted by: *{bread, cheese} ⇒ {milk}*

APRIORI ALGORITHM

The apriori algorithm is used for mining frequent itemsets from a dataset, from which association rules can then be derived. It is based on the intuitive **apriori principle:**

If an itemset is infrequent, then any larger itemset containing that itemset is also infrequent.

Using this principle, the algorithm works via an iterative procedure of "pruning" any candidate itemsets below the chosen **support threshold** - starting with single items, and then building double itemsets, triple itemsets and so on, until no itemsets pass the threshold. The rules can then be derived from the frequent itemsets by splitting into a LHS and RHS.

Terminology / Labels:
Below is a list of terms that are used to denote itemsets

"Itemset X" can also be written as:	*"Itemset Y"* can also be written as:
• LHS (left-hand side) • Antecedent	• RHS (right-hand side) • Consequent

There are many metrics that can be used to determine the strength and usefulness of associations between itemsets, including **support**, **confidence**, **lift**, **leverage**, and **conviction**.

SUPPORT

Support represents the popularity of an itemset. The support of itemset X is denoted by *support(X)*, and is simply the proportion of transactions in the database in which the itemset X appears. It is calculated as the following:

$$Support(X) = \frac{Number\ of\ transactions\ in\ which\ X\ occurs}{Total\ number\ of\ transactions}$$

The frequency of itemset *X* occurring will never be greater than the total number of transactions. Therefore, the support of an itemset always be between 0-1.

PRACTICE QUESTION:

1. Evaluate the following from the sample dataset used above:
    a. Support (Milk)
    b. Support (Milk and Cereal)
    c. Support (Milk and Cheese)
    d. Support (Milk, Cheese, and Bread)

CONFIDENCE

The confidence of a rule represents the likelihood of itemset Y being present given that itemset X is present. It is calculated as the following:

$$Confidence\ (X => Y) = \frac{Support(X\ \&\ Y)}{Support(X)}$$

- *Support(X & Y)*: The support of itemsets X *and* Y occurring together in a single transaction
- *Support(Y)*: How often itemset Y occurs on its own

One major drawback of confidence is that it can be large simply by virtue of the fact that itemset Y is popular (i.e. itemset Y on the RHS is inherently popular, irrespective of itemset X on the LHS). E.g. Confidence(Jalapenos -> Milk) may be high just because everyone buys milk (including those people who buy Jalapenos), irrespective of this having anything to do with Jalapenos.

A note on confidence:

Confidence is akin to *conditional probability* i.e. the probability of event Y given that event X has occurred, written as P(Y|X). Confidence is always between 0-1.

PRACTICE QUESTION:

1. Consider the following from the example dataset used above:
    Confidence (Milk ⇒ Cheese)
    Confidence (Cheese ⇒ Milk)
    Confidence (Milk ⇒ Cereal)
    a. What can you infer from these values?
    b. Does the order of the items matter when calculating confidence?

LIFT

Lift measures the strength of the association rules in question, and overcomes the drawback of confidence. It does this by measuring how much the presence of one itemset (e.g X) affects or 'lifts' the presence of the other itemset in the rule (e.g. Y). In other words, does the presence of itemset X increase the presence of itemset Y beyond chance (i.e. compared to if they were independent)?

Lift is calculated by comparing the support of itemsets X and Y *together* with that of itemsets X and Y *alone*.

$$Lift\,(X => Y) \;=\; \frac{Support(X \,\&\, Y)}{Support(X) \times Support(Y)}$$

Because lift is in essence a ratio, it can take on any positive value from 0 to infinity (∞).

Values of Lift		
Lift < 1 Itemsets X and Y are *less* likely to occur together than independently of each other; the presence of one itemset negatively affects the presence of the other. Strong negative lift would have a value close to 0.	*Lift = 1* Itemsets X and Y are *independent* of each other, and are *equally* likely to be bought together as they are to to be bought separately. No interesting association can be derived.	*Lift > 1* Itemsets X and Y are *more* likely to occur together than independently of each other; the presence of one itemset "lifts" (positively affects) the presence of the other. Strong positive lift could be any value greatly above 1.

PRACTICE QUESTION:

2. Evaluate the following using the example dataset above:
   Lift (Milk ⇒ Cheese)
   Lift (Milk ⇒ Cereal)
   Lift (Cereal ⇒ Milk)
   a. What can you infer from these values?
   b. Does the order of the items matter when calculating lift?

LEVERAGE

Similarly to lift, leverage looks at the difference between the support of two itemsets occurring together and the support of the two itemsets being independent from each other.

Unlike lift, it calculates this difference by subtracting, rather than dividing, the independence of itemsets X and Y. The equation is written as follows:

$$Leverage\,(X => Y) \;=\; Support(X\&Y) - (Support(X) \times Support(Y))$$

The values of leverage can range from -1 (negatively associated) to 1 (positively associated).

Values of Leverage		
Leverage < 0 Itemsets X and Y more often occur *independently* than together. The minimum leverage (most negative association) = -1.	*Leverage = 0* Itemset X and itemset Y are *independent* of each other, and are *equally* likely to occur together as they are to occur separately.	*Leverage > 0* Itemsets X and Y more often occur *together* than independently. The maximum leverage (most positive association) = 1.

PRACTICE QUESTION:

3. Evaluate the following using the example dataset above:
   Leverage (Milk ⇒ Cheese)
   Leverage (Milk ⇒ Cereal)
   Leverage (Cereal ⇒ Milk)
   a. What can you infer from these values?
   b. Does the order of the items matter when calculating leverage?

CONVICTION

Conviction measures the *dependence* of itemset X on itemset Y. A high conviction value means that itemset Y is highly dependent on itemset X:

$$Conviction\ (X => Y) = \frac{1 - Support(Y)}{1 - confidence(X => Y)}$$

Conviction values can range from 0 to infinity (∞).

Values of Conviction	
Conviction = 0 ● *Support(Y)* = 1, so the numerator (and thus conviction) equals zero ● Therefore, conviction of zero means that itemsets X and Y are independent	*Conviction = ∞* ● *confidence(X⇒Y)* = 1, so the denominator becomes zero ● Therefore, there is infinite conviction that itemset Y depends on itemset X

PRACTICE QUESTION:

4. Evaluate the following using the example dataset above:
   Conviction (Milk ⇒ Cheese)
   Conviction (Milk ⇒ Cereal)
   Conviction (Cereal ⇒ Milk)
   a. What can you infer from these values?
   b. Does the order of the items matter when calculating conviction?

3. LINEAR REGRESSION

OVERVIEW

This summarises different metrics of linear regression (r^2, error measurements). In brief, r^2 is a measure of fit explaining how well the input variables explain the variation in the output variables. Residuals are the differences between the actual values in the data and the estimated values derived from a model.

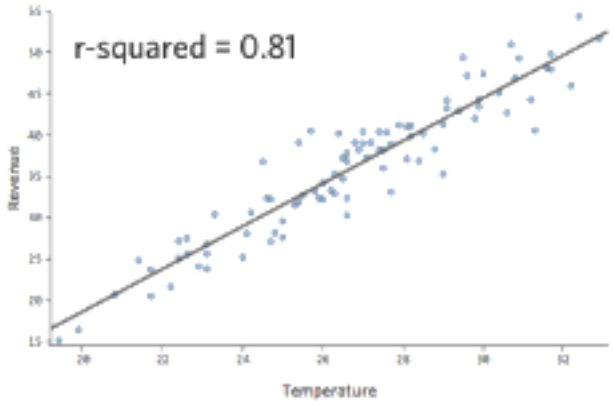
KEY CONCEPTS & TERMINOLOGY

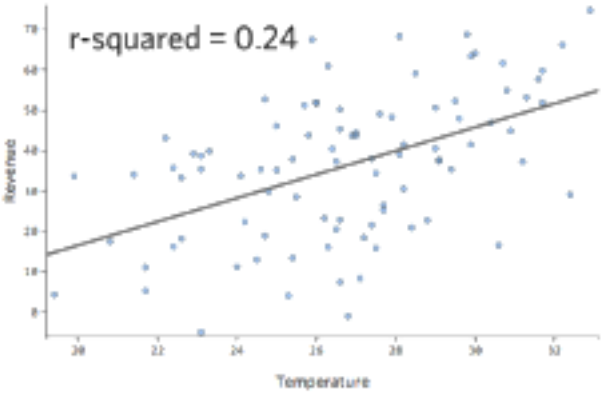
Coefficient of Determination (r^2 value)	Residuals
Mean Absolute Error (MAE)	Mean Square Error (RMS)
Root Mean Square Error (RMSE)	

THE COEFFICIENT OF DETERMINATION (r^2 VALUE)

The value of r^2 is a measure of fit and explains how closely data fits a regression line. It is a proportion ranging from 0 to 1, with higher values (close to 1) indicating better model fit, and lower values (closer to 0) indicating worse model fit.

Meaning of r^2		
Mathematically, r^2 is the extent to which the output variance (Y) is explained by the input (X)	Larger r^2 values indicate a larger amount of the variation in Y can be explained by the presence of X	The r^2 value can be converted to a percentage by multiplying it by 100





Limitations of r^2	
It cannot tell if a model is ‘good’ or ‘bad’ • A good model can have a low r^2 • A bad model can have a high r^2	It cannot tell if the data and/or predictions are biased in any way

PRACTICE QUESTIONS:

1. In the two graphs above, which appears to have a better fit? Why?

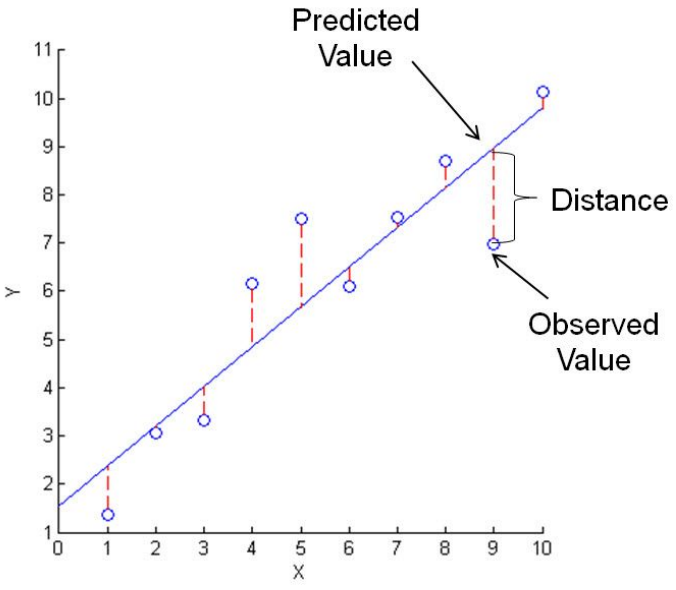
2. If X accounts for all of the variation in Y (i.e., all the data points fall exactly on the regression line), what would the r^2 value be?

3. If X accounts for none of the variation in Y (i.e., the regression line is perfectly horizontal), what would the r^2 value be?

RESIDUALS

Residuals are the difference between actual (observed) values in the dataset, and the model's estimated (predicted) outputs. They are calculated for every data point in the dataset. You can use a model's residuals to judge the usefulness of a model.

With large sets of data, residuals are difficult to inspect individually, so there are many summary statistics that condense all the residuals into a single value to assess a model. Examples include MAE, MSE, and RMSE. ***Note:*** the word *'error'* often refers to *residuals*.

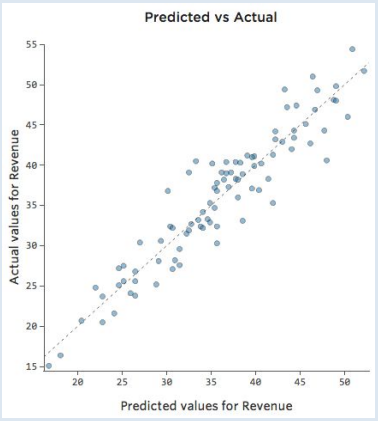


Residual values can be used to determine if a model is good at predicting the output	Small residual values imply the model may be a good predictive model	Large residual values imply the model may be a poor predictive model

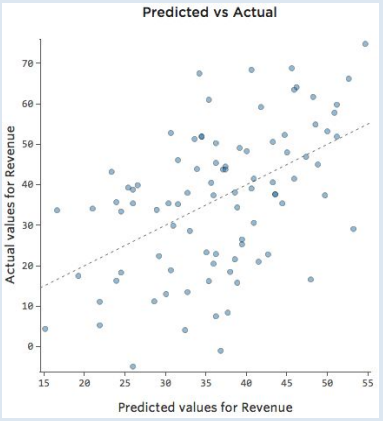
PRACTICE QUESTIONS:

Look at the graphs below:

Graph A



Graph B



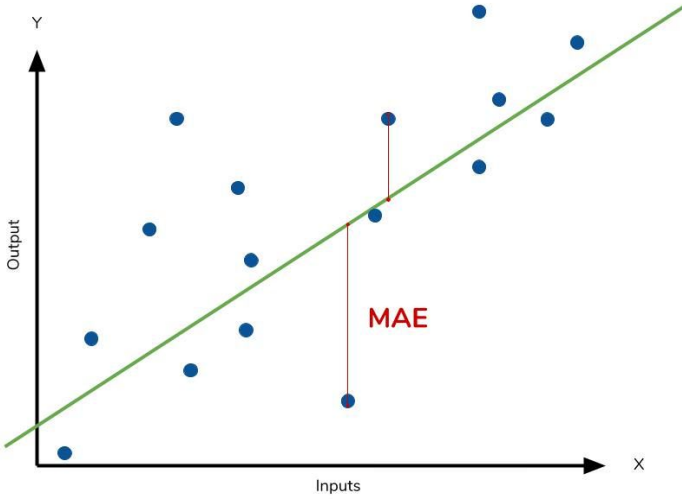
4. Based on the residuals in the graphs above, which one appears to be a better-fitting model? Why?

RESIDUALS: MEAN ABSOLUTE ERROR (MAE)

MAE is the simplest residual measurement to measure *accuracy* for continuous variables

With MAE, *Absolute values* (positive) of residuals are taken, then summed to take the mean - MAE values can range from 0 to infinity

Each residual in MAE contributes proportionally to the total error of the model

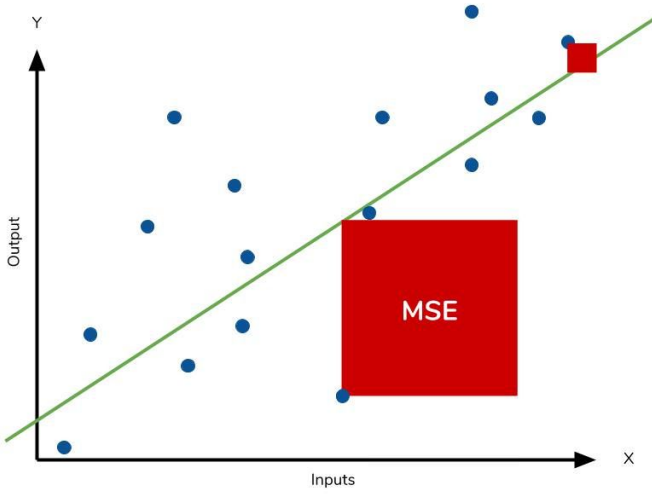


RESIDUALS: MEAN SQUARE ERROR (MSE)

With MSE, the residuals are squared, and then summed to take the mean - the value of MSE range from 0 to infinity

Because the residuals are squared, MSE will almost always be bigger than MAE, and so they cannot be directly compared

With MSE, the error values grow quadratically, not proportionally, so any outliers in the data will contribute to a higher total error to the model than with MAE errors



RESIDUALS: ROOT MEAN SQUARE ERROR (RMSE)

RMSE is a measure of how large the residuals are spread out in a regression model	Squaring of residuals (as in MSE) leads to units that do not match the original output units (e.g., kg to kg^2, or £ to $£^2$) making interpretation hard	RMSE converts the units back into something interpretable (i.e., back to their original form), by taking the square root of MSE

PRACTICE QUESTION:

5. How does MSE contrast with how MAE residuals are measured?

6. What is the benefit of using RMSE as opposed to MSE?

4. LOGISTIC REGRESSION (CLASSIFICATION)

OVERVIEW

Logistic regression is a type of regression that can actually be used for classification problems. A useful read about logistic regression versus other classification methods can be found here.

KEY CONCEPTS & TERMINOLOGY

Binary Logistic Regression

Other Types of Logistic Regression

Probability Threshold

BINARY LOGISTIC REGRESSION

One extension of regression models is logistic regression, which can be used for solving classification problems. Binary logistic regression is one type of logistic regression, where the outcome is dichotomous (or binary).

Features:	***Examples:***
Used when the *dependent* (target) variable is dichotomous, or 'binary'	'Yes' or 'No' as an answer to the question: "Did you vote in the last election?"
Independent variables (features) can be any data type: nominal, ordinal, interval or numeric. *Caution(!):* Adding too many independent variables can lead to overfitting the model	Variables such as age, income, what area they live in, their political preferences, etc.
Used to see how independent variables affect the likelihood (or probability) of being in one category or the alternative category (real-life example here)	How age, income, residential area, and political preferences affect the probability of them either voting ('Yes') or not voting ('No') in the last election
Outcome is the probability of being in a certain category (value ranging from 0-1)	

ASSUMPTIONS

1. The dependent variable is *dichotomous/binary*: e.g. 'yes' vs. 'no', 'present' vs. 'absent'
2. No outliers can be present in the data
3. There cannot be high correlations between features
4. Observations have to be independent of each other:
5. Observations should not come from repeated measurements over time (e.g., asking the same person every year if they voted or not)
6. Observations should not come from 'matched' data (e.g., measuring something about a person before and after they voted)

OTHER TYPES OF LOGISTIC REGRESSION

These have the same principles as binary logistic regression, but the dependent variable can have three or more categories as the outcome as opposed to a binary outcome.

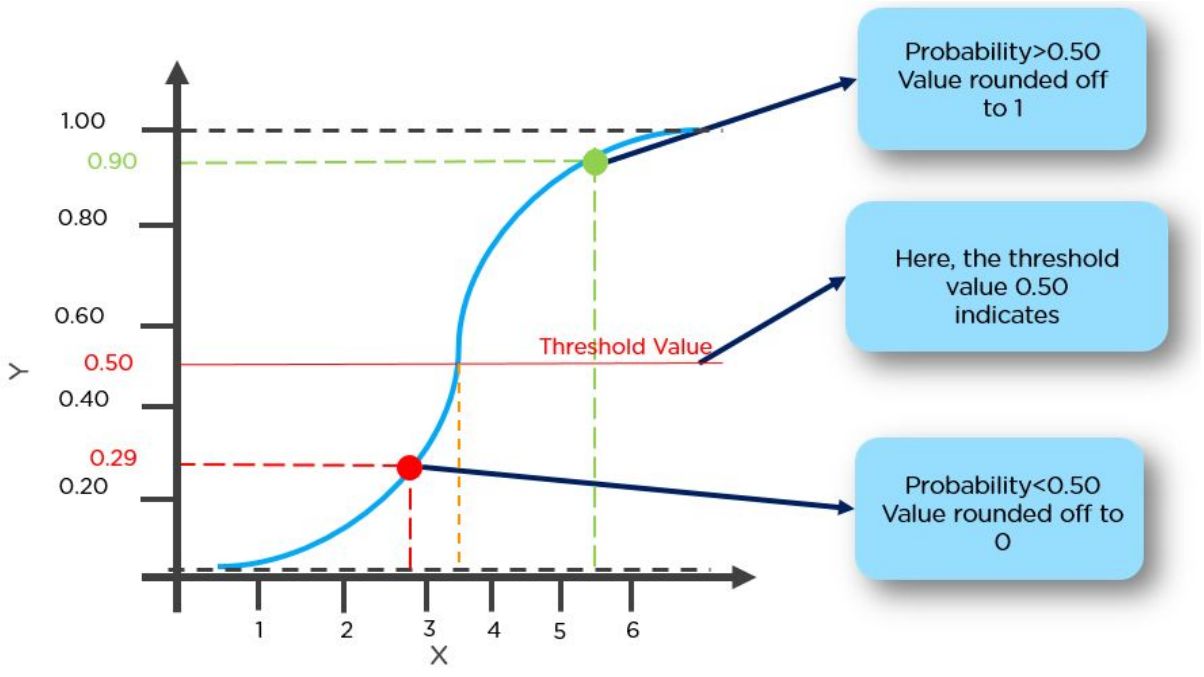
Multinomial logistic regression	***Ordinal logistic regression***
The categories in the dependent variable have no particular order to them, such as automotive types (e.g., train, car, or bus)	The categories in the dependent variable have a certain order to them, such as with ratings (e.g., very bad, bad, neutral, good, very good)

PROBABILITY THRESHOLDS

Logistic regression outputs follow a sigmoid (curved, S-shape) line, lying between 0-1 on the y-axis, with an assigned 'threshold' on the y-axis being the division point for two groups (see the red solid line in the image below).

This classification or probability threshold is a metric ranging from 0-1 that provides the model with a probability cutoff at which the data will be assigned to one category over another category (this website has a great illustrative explanation).

The default probability threshold is 0.5	
1. *If a value falls below 0.5:* The value is assigned to category '0' of the dependent variable (e.g., "No")	2. *If a value falls above 0.5:* The value is assigned to category '1' of the dependent variable (e.g., "Yes")



PROBABILITY THRESHOLDS

The Logistic Regression outcome is a probability score:		
Below threshold: If the value is below a certain threshold, the value is rounded down to 0, and assigned to the category that has been labeled as '0'	*Above threshold:* If the value is above a certain threshold, the value is rounded up to 1, and assigned to the other category that has been labeled as '1'	*Result:* The result is the classification of values to either one group or the other
The probability threshold is adjustable:		
You can adjust the threshold in order to optimise the division of the groups, and maximise sensitivity and specificity for classifying the groups.	The sigmoid 'S' graph with a threshold *other than 0.5* will look different, with the more 'vertical' part of the 'S' curve either higher or lower.	ROC curves can help with choosing the most appropriate probability threshold. Larger area under the ROC curve (AUC), means a better model.

INTERPRETATION OF RESULTS

The logistic regression model is defined using the sigmoid or logistic function:

Linear Regression Equation	***Logistic Regression Equation***
$$y = \alpha + \beta X$$ *Parameters are fit using ordinary least squares (OLS)*	$$y = \frac{e^{\alpha+\beta X}}{1 + e^{\alpha+\beta X}}$$ *Parameters are fit using maximum likelihood estimation (MLE) or some other algorithm*

Note that, unlike in linear regression, a one unit increase in X will *not* be associated with a β unit increase in Y. Instead, you would say a one-unit increase in X multiplies the odds by e(β). This is because the slope of the logistic regression curve is *not constant or linear*; rather it flattens for small or large values of X's. For more information on interpreting outputs, please read this link.

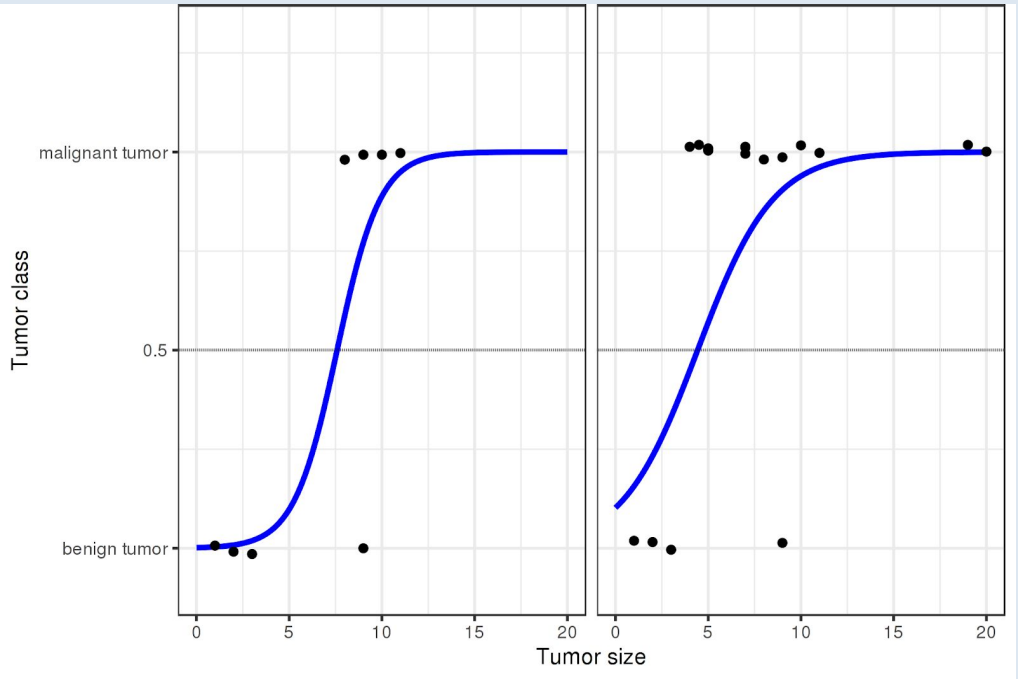
ADVANTAGES & DISADVANTAGES OF LOGISTIC REGRESSION

Advantages	***Disadvantages***
1. A type of predictive classification model 2. Probabilities as outputs can be advantageous (e.g., knowing a class occurrence has a 99% probability compared to 51%) 3. Can be used with binary or categorical dependent variables	1. Less predictive power that other classification models 2. Interpretation of the outcome can be difficult 3. In situations where the data separates into two groups perfectly, you cannot train the model

PRACTICE QUESTIONS:

1. Identify whether these dependent variables are binary, multinomial, or ordinal:
    a. “Favourite colour”: ‘Red’, ‘Blue’, ‘Green’, ‘Yellow’
    b. “How do you feel?”: ‘very cold’, ‘cold’, ‘cool’, ‘average’, ‘warm’, ‘hot’, ‘very hot’
    c. “Type of tree”: ‘deciduous’, ‘evergreen’

See the image below, and use to answer questions 2-4:



2. What is the probability threshold?

3. What are the two categories the threshold is separating?

4. Which graph has a better division of categories - the left or the right? Why?

5. NAIVE BAYES CLASSIFIERS (CLASSIFICATION)

OVERVIEW

This summary discusses Naïve Bayes algorithms including theorem overview, classifiers, diagnostic criteria, and corrections/smoothing. If you feel you need more information, here is a helpful resource.

KEY CONCEPTS & TERMINOLOGY

Bayes' Theorem

Classifiers & Distributions

Laplace's Correction (Smoothing)

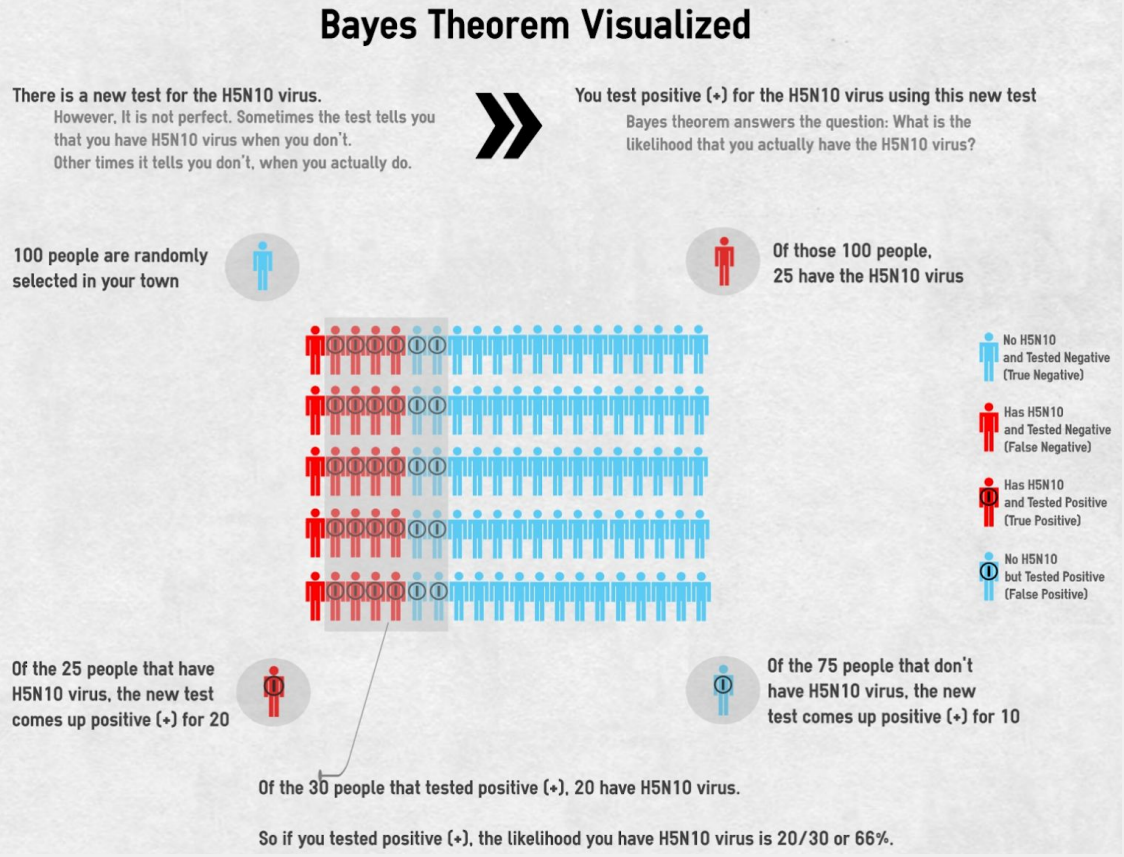
BAYES' THEOREM

Bayes' theorem is related to the concept of conditional probability - the probability of event 'A' occurring given that event 'B' has already occurred.

The image below is an example of Bayes theorem, applied to testing people for a virus.

In a town of 100 people, some of them are infected with a virus (red), and some are not (blue). Based on a previous event (being infected with the virus, 'B'), we want to predict the probability of someone showing the virus in a new test (our event 'A').

In this case, the theorem attempts to predict the probability of someone with the virus also being tested as having the virus.



BAYES' THEOREM: THE EQUATION

The equation is $P(A|B) = [P(B|A)*P(A)] / P(B)$ where:

Event 'A' is the 'hypothesis', or the class variable, for which we calculate the probability.	Event 'B' is the 'evidence', or the predictor we assume to have occurred (i.e., the characteristic of the object).	P(B) cannot equal zero.

APPLICATION

- Spam filtering for emails
- Text Analysis
- Recommendation systems
- Sentiment analysis
- Predicting defaults in e-lending
- Document type classification

LIMITATIONS

Below we will discuss the Naïve Bayes classifier, which has one main limitation - it requires that predictor events ('B') are independent, and are discrete values. However, in real-life situations, these predictor events are normally dependent with respect to new events ('A'). Therefore, its application towards the real world might not always be apparent in some cases.

NAÏVE BAYES CLASSIFIERS AND DISTRIBUTIONS

Naïve Bayes is the application of Bayes' Theorem towards determining the probability of an event occurring, given that we know the prior probabilities of past *independent* events.

Bayes' Theorem is used to make **classifiers**: models that assign class labels to instances represented as feature values ('B'), where the class labels are drawn from a set of information to then assign to a new hypothetical event ('A').

Naive Bayes assumes that classifier events are all independent:

Imagine you have some *independent* feature variables B_1, B_2, etc. and a class variable A.

The naïve Bayes prescription says to compute the following conditional probability (which in this context is called the **score**):

$$P(A \mid B_1, B_2 ...) \sim P(B_1 \mid A)*P(B_2 \mid A)* ... *P(A)$$

where the "**...**" means include any other feature variables that are present, and the ~ means that this is a proportional relationship rather than direct equality. There should be a term in the denominator, but it only depends on the features, and so is the same for each class value - it thus doesn't affect our class prediction.

The fact that the conditional probability simplifies to this is because we assume the features are independent.

The assumptions used on the distributions of the features depend on the problem at hand - these are used to calculate the prior probabilities.

Additional Naive Bayes Distributions		
Gaussian naïve Bayes: Assumed that predictors take up a continuous value (are not discrete), and have normal distributions.	*Multinomial naïve Bayes:* Mostly used for discrete counts where the predictors used by the classifier are the documents' word frequencies.	*Bernoulli naïve Bayes:* Similar to multinomial, but the predictors are binomial Boolean (yes/no) variables (e.g., if the word occurs or does not occur in the text).

NUMERICAL UNDERFLOW

When you have a large number of features (or features with many levels), the **scores** can become very small in magnitude (i.e. close to zero) - this problem is called numerical underflow, and is caused by multiplying several probability values that are close to zero. One way to alleviate this is to compute the *logarithm* of the products, which is equivalent to the summation of the logarithm of the probabilities. Implementations of Naive Bayes usually do this regardless of how many features there are, just to make sure numerical underflow doesn't occur.

LAPLACE'S CORRECTION (SMOOTHING)

As both prior P(A) and conditional P(B|A) probabilities are estimated using training data, they are subject to errors. Laplace's correction is a technique used to smooth categorical data by adding '1' to every feature count in the data.

All values (both null and non-null) are increased by a value of 1	Adding '1' helps to avoid instances of calculating the probability of an event 'B' to be 0 (i.e. an impossible event)	The fact you are testing for an event means it is not impossible (and therefore not 0) - adding 1 adjusts this to a small, arbitrary number in order to perform calculations

One situation where a zero count could be problematic is when an event in the testing data might appear, but for some reason the same event was not present in the training data. This would lead to a situation where posterior probability is 0 due to the random splitting of the training/testing datasets, even though in total, the event does occur.

In other situations, you might choose to drop any values with a 0 count and not use smoothing at all in order to preserve the "trueness" of the existing values.

PRACTICE QUESTIONS:

1. In your own words, what is the Bayes Theorem testing?
2. Note down some additional examples of the above applications of Bayes Theorem
3. Why would you perform Laplace smoothing?
4. If you wanted to analyse text documents, what type of naïve Bayes distribution would you choose?

6. DECISION TREES (CLASSIFICATION)

OVERVIEW

Here we will discuss the use of Decision Trees as a method of classification modelling, as well as how a confusion matrix, ROC curve, and AUC are used to assess the resulting model. For additional information, please visit this website for more example diagrams and uses.

KEY CONCEPTS AND TERMINOLOGY

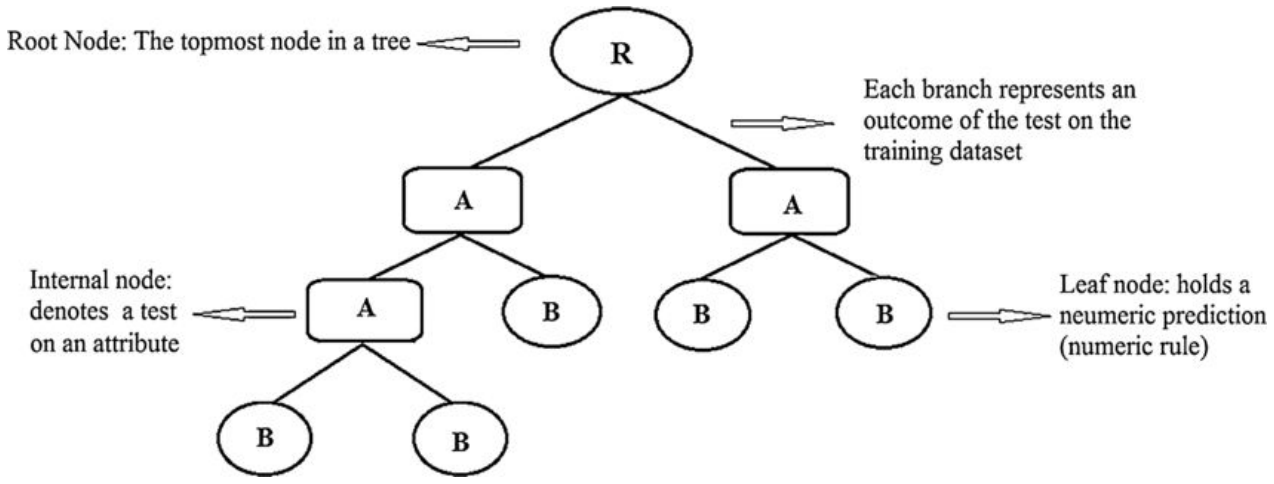
Decision Trees	Entropy/Information Gain
Gini Index	Confusion Matrix & Metrics
ROC Curve	AUC

DECISION TREES

Decision trees use tree-like models to make decisions based on many features, and then determine the possible outcomes. They are great for predictive models, having high accuracy, stability, and ease of interpretation.

Unlike linear models (e.g. linear regression), they can map non-linear data relationships quite well. However, they can suffer from overfitting or class bias, can lose information when using continuous data, and are sensitive to small changes in the data.

In the below figure, each **node** represents a "question" on an attribute (e.g. whether or not a car is red). The very first node is called a "**root node"**, while all subsequent nodes up until the end are called **"internal nodes"**. **Branches** represents the outcome of each node (e.g., yes or no) that connect to the next node. The final node - or **"leaf/terminal node"** - has the final class label and outcome prediction. If you have too many nodes, removing extra levels is called **"pruning"**.



Types of Decision Trees	
Categorical: attempts to answer a question that has a *categorical* target variable (classification). For example, if the question (variable) you are attempting to answer has an outcome that is categorical: “yes” vs. “no”; “high-risk” vs. “low-risk”; “fraud” vs. “not fraud”; etc.	*Continuous:* attempts to answer a question that has a *continuous* target variable (regression). For example, if the question (variable) you are attempting to answer has an outcome that is numeric: age, house price etc.

In both cases above, the *feature* variables can be a mix of categorical and numerical.

GINI INDEX VS. ENTROPY/INFORMATION GAIN

To split the data at a specific node, different algorithms can be used depending on whether they have categorical or continuous outcomes. For trees with categorical outcomes, most algorithms make use of **Gini Index** or **Entropy** - two different measures of “purity” of a set of data points.

Split Algorithms for Categorical Data	

Gini (CART):
• Higher gini score indicates higher purity (sameness) of a node:
  ○ Pure node: gini = 0
  ○ Maximally impure node (i.e. 50-50 split of data): gini = 0.5
• Choose the split condition that *maximises the difference in gini* before and after the split
	Entropy/Information Gain (ID3/C4.5):
	• Higher entropy score indicates higher purity (sameness) of a node:
	  ○ Pure node: entropy = 0
	  ○ Maximally impure node (i.e. 50-50 split of data): entropy = 1
	• Choose the split condition that *maximises the difference in entropy* before and after the split - this difference in entropy is called **information gain**.

CONFUSION MATRIX & METRICS

A confusion matrix (or “error matrix”) describes the classifier performance on a set of test data, for which the true values are known. It classifies which data has been correctly or incorrectly classified as positive (true) or negative (false). It gives us insight into the errors being made by a classifier, and the types of errors that are being made.

	YES (predicted)	**NO (predicted)**
YES (actual)	True Positive (TP)	False Negative (FN)
NO (actual)	False Positive (FP)	True Negative (TN)

Metrics can be calculated using a confusion matrix to assess model classification (seen below):

Confusion Matrix Metrics		
Accuracy: How many data points were correctly identified by the model as being either true or false.	***Precision (Positive Predicted Value):*** From the total number of data points *predicted* to be *true* by the model, how many were actually classified as true in the dataset.	***Recall/Sensitivity or True Positive Rate (TPR):*** From the total number of data points *actually* classified as *true* in the dataset, how many did the model *correctly predict* as *true*.
Equation: $\frac{(TP + TN)}{(TP + TN + FP + FN)}$	*Equation:* $\frac{TP}{(TP + FP)}$	*Equation:* $\frac{TP}{(TP + FN)}$
Specificity: From the total number of data points *actually* classified as *false* in the dataset, how many were *correctly predicted* to be true by the model.	***False Positive Rate (FPR):*** From the total number of data points *actually* classified as *false* in the dataset, how many were *incorrectly predicted* to be true by the model. (Also known as 1-specificity)	***F-measure (F1):*** A weighted average of precision and recall. Therefore, it accounts for both false positives and false negatives in the equation. More useful than accuracy if class distribution are uneven
Equation: $\frac{TN}{(TN + FP)}$	*Equation:* $\frac{FP}{(FP + TN)}$	*Equation:* $\frac{(2*Recall*Precision)}{(Recall + Precision)}$

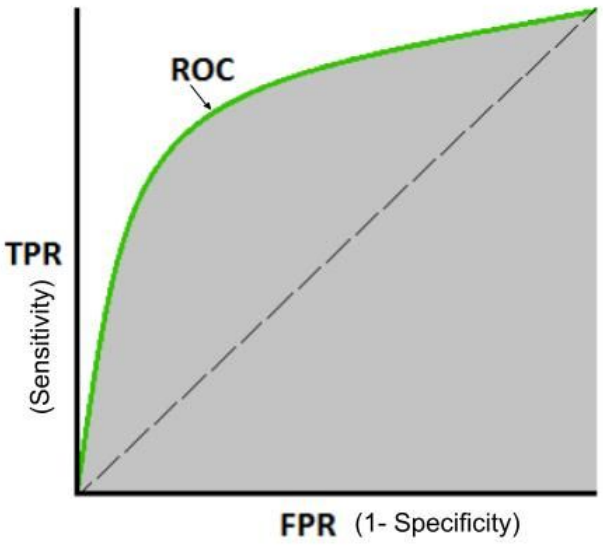
ROC CURVE

An ROC curve (receiver operating characteristic curve) is a graph showing the *performance of a classification model* at all classification thresholds. This curve plots the following two parameters against each other:

- True Positive Rate (TPR); also called Recall or Sensitivity
- False Positive Rate (FPR); also called 1-specificity

Below is a typical ROC curve. Notice how the green line goes up fairly soon and then slowly tapers off. Ideally, a good classification model attempts to maximise TPR (sensitivity), and minimise FPR (1-specificity), as demonstrated by the green curve.

ROC curve: classification thresholds
Increasing the threshold: Classifies *less* total items as positive; so there are fewer True Positive values and False Positive values; better division of classes
Decreasing the threshold: Classifies *more* total items as positive; so there are more False Positive values and True Positive values; worse division of classes



AUC: ASSESSING THE ROC CURVE

The AUC (or "Area Under the ROC Curve") measures the entire area under the ROC curve. The value of the AUC provides an overall measure of performance across all possible classification thresholds. AUC values range from 0 to 1.

AUC Values	
AUC = 0.0 A model has 100% of its predictions *wrong*	*AUC = 1.0* A model has 100% of predictions *correct* (at a particular threshold value)

A classification model can have either a good ROC curve (and thus a high AUC), or a bad ROC curve (and thus a low AUC), depending on the data being used.

AUC: ADVANTAGES AND DISADVANTAGES

Advantages:	***Disadvantages***:
1. Measures how well the predictions are *ranked*, rather than their values 2. Measures the *quality* of the model's predictions *regardless* of what classification threshold is chosen	1. Might not always want a measure of prediction ranks; sometimes you might need the actual probability outputs 2. You might not always want to assess the overall quality of model predictions where there are different *practical consequences* in whether you prioritise having higher false positive or false negative rates

PRACTICE QUESTIONS:

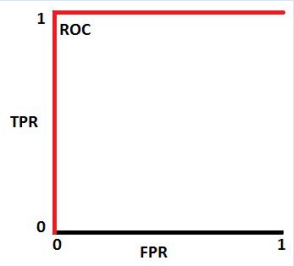
1. Using the Confusion Matrix below, calculate the following values:

	YES (predicted)	NO (predicted)
YES (actual)	100	10
NO (actual)	20	50

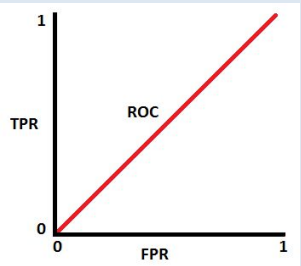
   a. Accuracy
   b. Recall
   c. Precision
   d. False Positive Rate

Look at the three graphs of ROC curves below to answer questions 2-3:

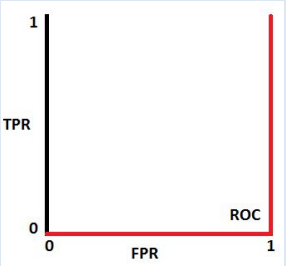
ROC curve A:



ROC curve B:



ROC curve C:



2. Match the ROC curves to the corresponding classification model where the probability threshold has a value of:
   a. Threshold = 0.95
   b. Threshold = 0.05
   c. Threshold = 0.50

3. Assume all three curves come from models with thresholds all equal to 0.5. Match the ROC curve to the model where the AUC has a value of:
   a. AUC = 0.5
   b. AUC = 0
   c. AUC = 1

4. What are the advantages and disadvantages of using decision trees?

7. TIME SERIES ANALYSIS

OVERVIEW

This is a summary about using time-based data to predict current observations using previous data. This section will focus specifically on a class of models called AutoRegressive Integrated Moving Average (ARIMA) models, which are useful for analysing and forecasting time series data. Read the information on this website for further information.

KEY CONCEPTS AND TERMINOLOGY

ARIMA Model	Autoregressive Model (AR)
Integrated Model (I)	Moving Average Model (MA)
Partial Autocorrelation Function (PACF)	Stationarity & Differencing
Autocorrelation Function (ACF)	Box-Jenkins Method
Akaike Information Criterion (AIC)	Additive/Multiplicative Model Decomposition

ARIMA MODEL

The ARIMA(p,d,q) model is a method to analyse time-based data. It uses a combination of **A**uto**R**egressive and **M**oving **A**verage models that deals with non-stationary data (data without measurable seasonality or trends). Since differencing is being used to make a time series stationary, **I**ntegration is part of the ARIMA model to replicate the original time series data.

AUTOREGRESSION MODEL (AR)

The 'AR' part of the ARIMA model describes the dependent *linear relationship* between a given observation, and any number of its prior (*lagged*) *observed values*. It states that for any given data point selected, it is dependent on its own previous **p** data points in the past.

INTEGRATED MODEL (I)

The 'I' part of the ARIMA model represents the *differencing* of raw data values to allow for the time series to become stationary. Raw data values are replaced by their *difference* between the data values. These differences are then modelled via the AR and MA models, and those results then integrated to replicate the original time series data.

MOVING AVERAGE MODEL (MA)

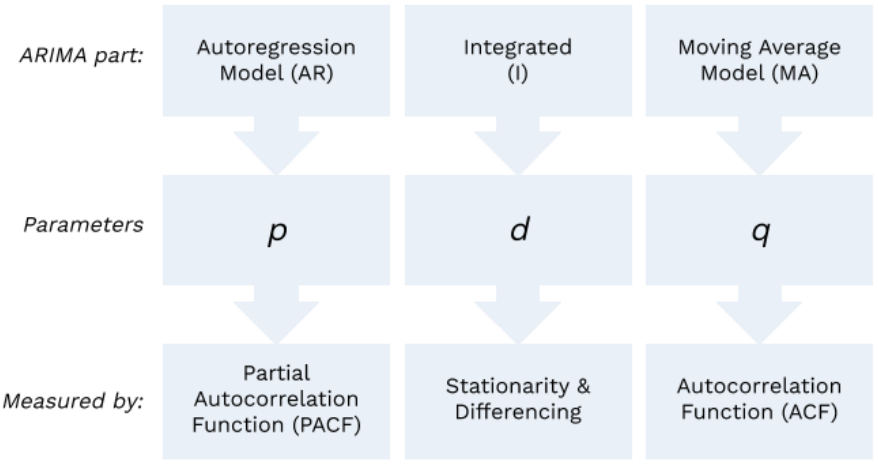
The 'MA' part of the ARIMA model incorporates the dependent *linear relationship* between an observation and **q** *lagged* (previous) *observed residuals* (error) in the past.

The residuals represent random occurrences (or "shocks"). The MA model determines how far into the past a 'random shock' (residual) still influences a present data point.

ARIMA MODEL PARAMETERS

The ARIMA model is represented as different parameters - p, d, and q. Each parameter is measured using a different technique. The diagram below identified the part of the ARIMA model, the parameter it corresponds to, and the technique used to measure these parameters.

ARIMA parameters
p - the maximum order for the AR
d - order of differencing to attain stationarity in the model
q - the maximum order for the MA



AUTOCORRELATION FUNCTION (ACF)

ACF is used to determine the value of '*q*' in the ARIMA model. It is an estimate of the correlation between a time series and a delayed ('lagged') version of itself. For stationary time series data, this gives a measure for how much any data point in the time series is correlated with its previous data point residuals (errors).

PARTIAL AUTOCORRELATION FUNCTION (PACF)

PACF is used to determine the value of '*p*' in the ARIMA model. It calculates correlation between two data points with a given lag (i.e., the difference in data points), after removing the correlation effects of the lags in between (via a linear regression model subtraction). For example, for daily (weekly) data a lag of two is comparing data points two days (weeks) apart.

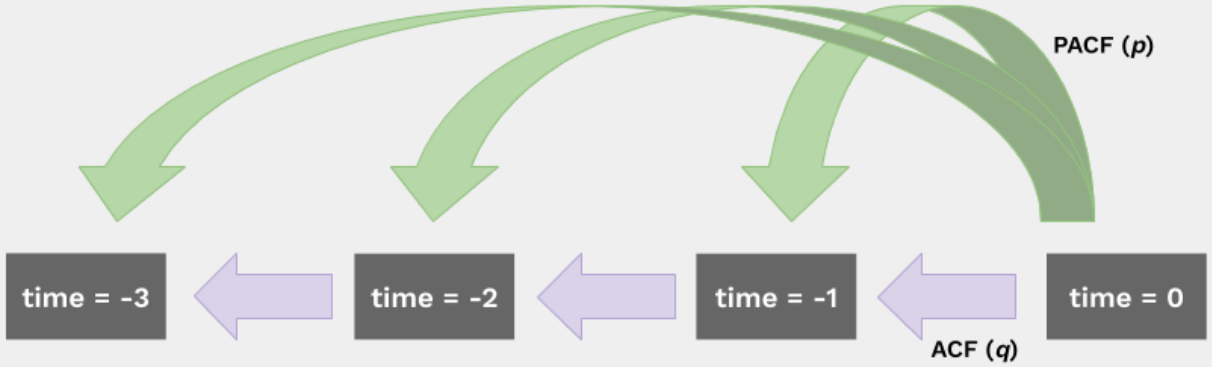
STATIONARITY & DIFFERENCING

If a time series is non-stationary (i.e., not constant over time), it can be turned into a stationary time series by applying a technique called *differencing*. The number of times we need to difference the data to make it stationary determines the parameter '*d*' in the ARIMA model.

Statistical tests for stationarity are for example the Augmented Dickey-Fuller test (null hypothesis: non-stationary, alternative: stationary), or KPSS Test (null hypothesis: stationary, alternative: non-stationary).

DIAGRAM OF ACF VS. PACF

Below is a visual representation of ACF (purple arrows) and PACF (green arrows). Starting at present time (time = 0), the ***ACF*** estimates how correlated that data point is with itself at a previous point in time, or lag (e.g., time = 0 to time= -1). This is repeated for each *lag* (e.g., time = -1 to time = -2; time = -2 to time = -3; and so on).



The ***PACF*** looks at the correlations between the present time (time = 0) and any other lag, removing the effects of any other time lags in between. For example, it looks at the correlation between time = 0 and time = -3, disregarding the effects of time = -1 and time = -2.

BOX-JENKINS METHOD

This is a standard approach for discovering the most appropriate values for *p*, *d*, and *q* in an ARIMA model. These 3 steps are repeated until the model achieves a good fit.

Steps of the Box-Jenkins Method:		
1. Model Identification Identify autoregressions and seasonality to determine the amount and size of the differencing and the lag.	*2. Parameter Fitting* Uses a fitting procedure to find the coefficients of the regression model.	*3. Model Evaluation* Tests residual errors to determine the amount and type of temporal structures not captured by the model.

ADDITIVE/MULTIPLICATIVE MODEL DECOMPOSITION

Time series data can be broken down into its basic components such as trend, seasonality, and residuals. An additive model decomposition would be appropriate if the fluctuations seem to behave independently of the trend.

If the fluctuations due to residuals and seasonal patterns behave proportionally to the underlying trend (i.e., higher fluctuations at higher values of the data and lower fluctuations at lower values), the appropriate decomposition would be multiplicative.

AKAIKE INFORMATION CRITERION (AIC)

The AIC is a measure for *model selection*, not model accuracy. The larger the number of parameters (model complexity), the larger the AIC. The better the fit of the model to the data, the lower the AIC. A lower AIC indicates the model is simpler (avoiding overfitting) and more accurate (avoiding underfitting) than other models.

PRACTICE QUESTIONS:

1. In the model ARIMA(2, 1, 3), what are the values of *q*, *d*, and *p*?
2. What is the difference between PACF and ACF?
3. What are the three steps of the Box-Jenkins Method?

8. TEXT ANALYSIS

OVERVIEW

This is a summary covering the different steps in text cleaning and analysis. This covers the various procedures needed to convert unstructured documents into clean, structured data (called *parsing*), and the types of analysis that can be done to gain insight from the text (e.g., sentiment and topic analysis).

KEY CONCEPTS AND TERMINOLOGY

Natural Language Processing (NLP)	Stop-words
Regular expressions (regex)	Tokenisation
Stemming	Lemmatisation
Bag-of-words	Sentiment analysis
Topic Analysis / Extraction	TF-IDF

NATURAL LANGUAGE PROCESSING (NLP)

Natural language processing (NLP) is a subfield of linguistics and machine learning. It deals with the interactions between computers and human languages - specifically, how to program computers to *process* and *analyse* large amounts of natural (human) language data. NLP techniques can include, but are not limited to:

- Tokenisation
- Stemming
- Lemmatisation
- Removing Stop-words
- Parts-of-Speech (POS) tagging
- Regular expressions

TOKENISATION

Tokenisation is the first step in text analysis. Here, paragraphs or chunks of text are broken down into smaller parts. These can be broken down into their individual sentences (*sentence tokenisation*) or individual words (*word tokenisation*). Once text is tokenised, you can also analyse *word frequency* to obtain a count of how often each word occurs in a document.

STEMMING

Stemming is a process of normalisation that reduces words to their main “root” word. Generally this happens by removing affixes (word beginnings) or suffixes (word endings). For example, the words “connection,” “connected,” and “connecting” all reduce to a common root word "connect". However, the stemming algorithm is fairly crude, and will remove letters from words that it thinks are suffixes, but not change the root accordingly. For instance, we know that “happy” is a root word of “happiness” and “happiest”. However, stemming will remove the -ness and -est endings, leaving you with a root of “happi” instead of “happy”.

LEMMATISATION

Lemmatisation is an extension of stemming, but takes into account the variations in spelling and format based on a word’s part of speech. It also reduces words to their base or root word, but

unlike in stemming, it also transforms the root words based on its usage (e.g., vocabulary, morphology, word tense).

It is generally more sophisticated than stemming. It works at an individual level with each word. For example, The words "better" “best” would both be lemmatised to "good"; “is”, “am”, and “are”, “were”, “was” would all be changed to the root verb “be”.

STOP-WORDS

Stop-words are the commonly-occurring words in a language. They tend to include prepositions, pronouns, articles, and certain verbs, including: is, am, are, this, a, an, the, them, etc. In text analysis, stop-words are considered noise as they do not contribute any meaningful information.

There is no one ultimate list of stop-words for removing them, but rather a variety of pre-made packages in many languages, each slightly different from one another. Not every unnecessary word will be removed, and sometimes you will have to remove them manually by creating your own stop-words list to use along with a predefined stop-words list.

PARTS-OF-SPEECH (POS) TAGGING

POS tagging is used to identify the grammatical group of a given word, whether it is a noun, pronoun, adjective, verb, adverb, etc. POS tagging looks at words in a sentence, and assigns the corresponding grammatical tags to words based on context, and relationships between words.

REGULAR EXPRESSIONS

Regular expressions (or “regex”) are sets of pattern-matching commands used to detect sequences of strings in text data. They can be used to search for and/or manipulate a variety of text objects, including:

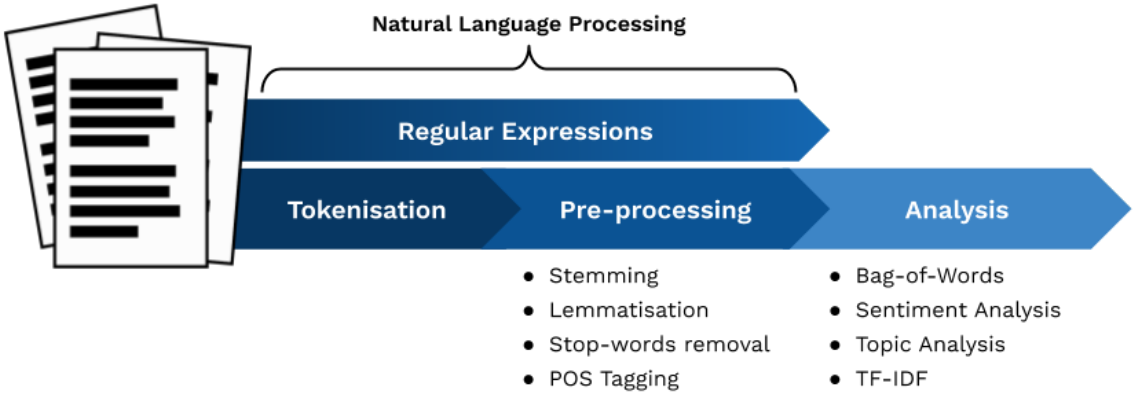
- Non-words within the text (e.g., spaces between words)
- Certain words or parts of words
- Metacharacters (examples of special characters: `\ . | ( ) [ ] { } $ * + ?`)

For example, the regex code `\w+` represents three things:

- “w” means word
- “+” means one or more (in this case, one or more words)
- “\” character is the escape character for regular expressions.

Using this regex and some additional functions returns anything that matches this regex criteria. For more information on list of regex character codes to use, please take a look at this resource.

The below diagram shows an overview of a generic text analysis workflow:



BAG-OF-WORDS

Bag-of-Words is a way to extract features from text to use in later machine learning modelling. It is a way to represent the frequency of words from a text document. It involves two things:

1. Vocabulary of known words
2. The counts of each individual known word

It's called a "bag" of words because any information about the order or structure of words in the document is removed. The model is only concerned with whether (and how often) a certain word *occurs* in the document, *not where* in the document.

Each individual word is given a score. These scores can be either boolean (i.e., 0 for not present, 1 for yes present), a count (number of times a word appears in the document), or a frequency (the ratio of each word count to the total word count in a document). The sets of words and associated scores are then saved as vectors.

While Bag-of-Words is simple to perform, flexible, and customisable, it still has some limitations, including vocabulary, sparsity, and meaning of words (see table below).

Limitations & Challenges of Bag-of-Words models:		
Vocabulary The vocabulary requires careful design in order to manage the size. This impacts the sparsity of the document representations.	*Sparsity* The sparse appearance of words are hard to model for both computational and information reasons. In these situations, there is very little information to model.	*Meaning* Bag-of-Words removes word order, which ignores context, and in turn meaning of words in the document (semantics).

SENTIMENT ANALYSIS

Sentiment analysis is the process of analysing and classifying *opinions* in text data. It has many practical applications, including the analysis of survey responses, product reviews, social media comments, and more.

Sentiment analysis can be applied to texts at different levels:

- Complete documents or paragraphs
- Individual whole sentences
- Parts of sentences/expressions
- Individual words

There are many kinds of sentiment, but some of the most commonly-used ones include:

- Polarity
- Emotion detection
- Aspect-based analysis
- Intent analysis
- Multilingual

POLARITY

The assignment of a numeric score - either positive, negative, or neutral - to each word. Some systems provide different scales of polarity by identifying positive or negative sentiment with a particular feelings; for example, anger or sadness as negative, and happiness as positive.

EMOTION DETECTION

This aims at detecting emotions like, happiness, frustration, anger, sadness, and the like. Many emotion detection systems resort to pre-defined lexicons (i.e., lists of words and the emotions they convey) or complex machine learning algorithms.

ASPECT-BASED ANALYSIS

Accounts for both the aspects or features of the text (the "what") as well as the polarity being used in the text.

INTENT ANALYSIS

Intent analysis basically detects what people want to do with a text rather than what people say with that text.

MULTILINGUAL ANALYSIS

Analysing text in many languages requires a lot of preprocessing using manually-created or online tools. A less complicated alternative would be to detect languages in the texts automatically, then train a custom model for the language of your choice, then perform analyses.

Limitations & Challenges of Sentiment Analysis:		
Tone & Subjectivity It can be difficult to detect subjective vs. objective texts. Many "objective" texts do not contain explicit sentiments.	*Context & Polarity* Everything is said in context, and analysing sentiment without context can be difficult. Context can change the polarity of a word, which can change the meaning.	*Sarcasm / Irony* Both irony and sarcasm can change positive sentiment into negative, and vice versa. Detecting sarcasm and irony requires contextual analysis, and thus hard to detect.
Comparisons The way in which comparisons are used is another challenge	*Emojis* Western and Eastern emojis use different text encodings that need to be handled differently. Emojis can be important with web-based text like twitter, so removal is not always ideal.	*Defining Neutral Polarity* Defining categories like positive, negative, and neutral can be subjective themselves. It is a very important step in sentiment analysis, especially when training a model.

TOPIC ANALYSIS / EXTRACTION

This is an NLP technique used to extract and understand what a given text is about. Topic analysis works by placing words into different "topics" based on the algorithm's determination of similarities or differences between words. You can think of topic analysis like a clustering algorithm for text, where each topic is a "cluster", and each word is a data point in that cluster. Topic analysis can be used in a variety of situations, from structuring and analysing customer feedback, to organising news articles according to their subject matter.

TERM FREQUENCY - INVERSE DOCUMENT FREQUENCY (TF-IDF)

One problem with scoring word frequency is that highly frequent words take over the document, but may not contain as much "informational content" to the model as rarer but perhaps domain-specific or important words. One way to overcome this issue is to rescale the frequency of words by how often they appear in all documents. This process is called Term Frequency – Inverse Document Frequency (**TF-IDF**).

In **TF-IDF**, words with larger frequency scores across all documents (which tend to be words of little informational value) are penalized, while words that appear less frequently are weighted higher. In other words, it is broken down into two parts:

- *Term Frequency*: scoring the frequency of a word in a single document
- *Inverse Document Frequency*: rarity of a word's appearance across all documents

Calculating Term Frequency-Inverse Document Frequency (TF-IDF):		
1. TF = $\frac{\#\ times\ a\ term\ is\ in\ a\ single\ document}{total\ \#\ terms\ in\ a\ single\ document}$	*2. IDF =* $log(\frac{total\ \#\ of\ documents}{\#\ times\ a\ term\ is\ in\ all\ documents})$	*3. TF-IDF =* $TF * IDF$

The resulting scores are a type of weighting, where not all words are equally as important or interesting. These scores highlight words that are distinct in a given document (and thus contain useful information), while dampening the effects of more-frequent, less-important words.

A TF-IDF score of 0 - the lowest possible score - indicates a high frequency of a word across documents, and therefore is likely to be unimportant. The higher the TF-IDF score, the less frequent a word is across all documents, and thus more likely to be an important term.

PRACTICE QUESTIONS:

1. What are the differences between tokenisation, stemming, and lemmatisation?
2. What is Sentiment Analysis, and what are the limitations?
3. Refer to the image below:

term	query			
	tf	df	idf	tf-idf
auto	0	5000	2.3	0
best	1	50000	1.3	1.3
car	1	10000	2.0	2.0
insurance	1	1000	3.0	3.0

   a. What term has the lowest document frequency? The highest document frequency?
   b. Based on the information provided, which term is likely to be the most "informative"? Why?

ADVANCED ANALYTICS: TECHNOLOGY & TOOLS

1. DATA WAREHOUSING

OVERVIEW

This summarises the concepts and tools of data warehousing and Big Data, including MapReduce and Hadoop environments and tools.

KEY CONCEPTS & TERMINOLOGY

Data Repository	Data Warehouse
MapReduce	YARN
Hadoop Ecosystem	NoSQL

DATA REPOSITORY

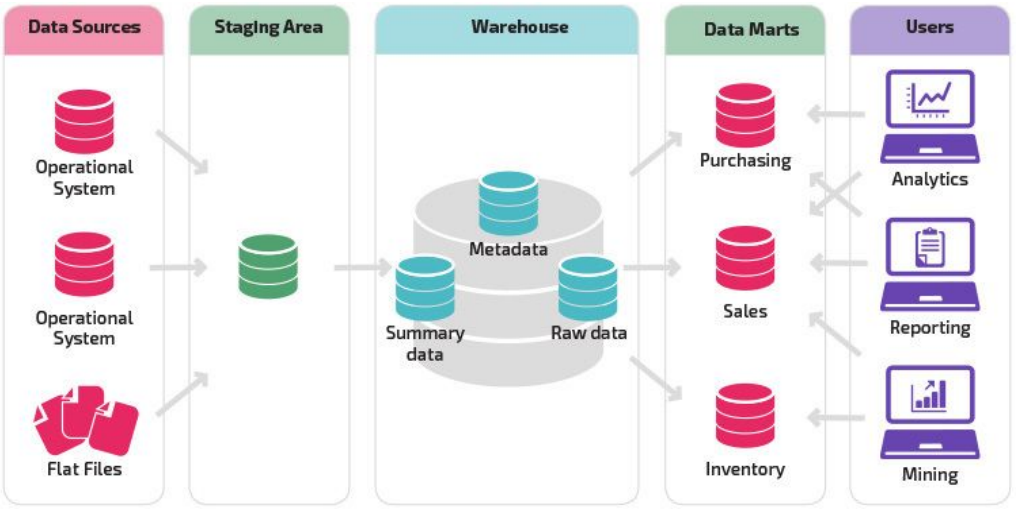
Data repositories describe the multiple ways to collect and store data. There are several types of data repositories, including **data warehouses**, **data lakes**, and **data swamps**:

Data warehouse	***Data lake***	***Data Swamp***
Large data repositories that aggregates *clean* and *sorted* (*structured*) data from multiple sources. The data is not all necessarily related to each other.	Large repositories of aggregated *unclean* and *unsorted* (and *unstructured*) data. The data analyst must clean and sort the data they need before analysis.	A type of data lake used to store all kinds of data but without any organisation or data management system. It can be very difficult to query and extract value from them.

DATA WAREHOUSE

Centralised repositories of data from one or more sources, placed in a purpose-built space. Analysts must access data remotely for extraction, transformation, and loading. Some characteristics of data warehouses include:

- ***Focus***: Enterprise-wide repository of disparate data sources
- ***Data Sources***: External and internal sources from different organisation areas
- ***Data Size***: 100 GB minimum, but often in the range of terabytes for large organisations
- ***Normalisation***: Modern warehouses are mostly denormalised for quicker data querying and read performance
- ***Decision Types***: Strategic decisions that affect the entire enterprise
- ***Cost***: Varies, but often greater than $100,000 (cloud solutions cost dramatically less)
- ***Setup Time***: At least a year for on-site warehouses (cloud warehouses are much quicker)
- ***Data Held***: Raw data, metadata, and summary data



DATA MARTS

Companies tend to store data in a single repository (Enterprise Data Warehouses – EDWs), where there is a single source of truth. Data lakes are common in start-ups and younger organisations. Once the data is stored in the EDWs, it can be stored in additional local systems in the form of departmental warehouses and *data marts*. Data marts are smaller, more bespoke, versions of data warehouses, and generally used for very specific purposes. They are created by business users to accommodate their needs for specific and deeper data analysis on a regular basis.

MAPREDUCE

MapReduce is a programming paradigm that allows for distributed computing based on Java. It allows for data processing to be broken down into smaller steps by using *parallel processing* techniques. The MapReduce process is broken down into 2 phases: “Map” and “Reduce”:

“Map” phase	***“Reduce” phase***
Applies operations to data, and provides intermediate output.	Consolidates intermediate outputs from “Map” phase to provide the final outputs.

YARN (YET ANOTHER RESOURCE NEGOTIATOR)

In the first version of Hadoop (v. 1.0), MapReduce not only handed the “Mapping” and the “Reducing” stages, but also the resource management of the cluster. When version 2.0 was released, these two activities were then separated -- MapReduce still handled distributed processing, but now, a tool called YARN was created for resource management. This ‘division of labour’ allows us the flexibility to use other paradigms instead of MapReduce if we so wish, without having to worry about resource allocation.

HADOOP ECOSYSTEM

Hadoop is a tool for data preparation and unstructured data analytics. It handles ***MapReduce*** activities, such as:

- Monitoring
- Scheduling
- Handling input, intermediate, and output steps

HDFS (HADOOP DISTRIBUTED FILE SYSTEM)

HDFS is a file system that provides the ability to store data in a distributed manner. It can perform these tasks by taking advantage of the parallel processing of MapReduce. Data is split and stored in HDFS blocks (or *chunks*) which usually store 64MB in each block, and stored across a cluster. When possible, the data chunks can also be stored across different machines in order to perform the "Map" steps of MapReduce in parallel. The HDFS filing system is achieved through three main components:

1. NameNode	***2. Secondary NameNode***	***3. DataNode***
A single machine responsible for tracking where each block of a data file is stored and replicated.	Helps the NameNode with certain tasks to reduce the load. However, it is *not* a backup node.	Stores and manages data in chunks. They also periodically send reports on the data they hold.

Because NameNode is a single machine, it creates a single point of failure for the entire HDFS system, which could lead to data unavailability. To counteract this limitation, it is usually run on a separate machine.

HDFS is also designed to counter faults by initially creating *3 copies of the same file* - all of which are redundant until the main blocks get corrupted or damaged. It is also designed to be 'rack-aware'. This means it would ideally distribute block copies across several equipment racks to defend against rack failure. For instance, if one rack completely burns down, we would still have backups elsewhere.

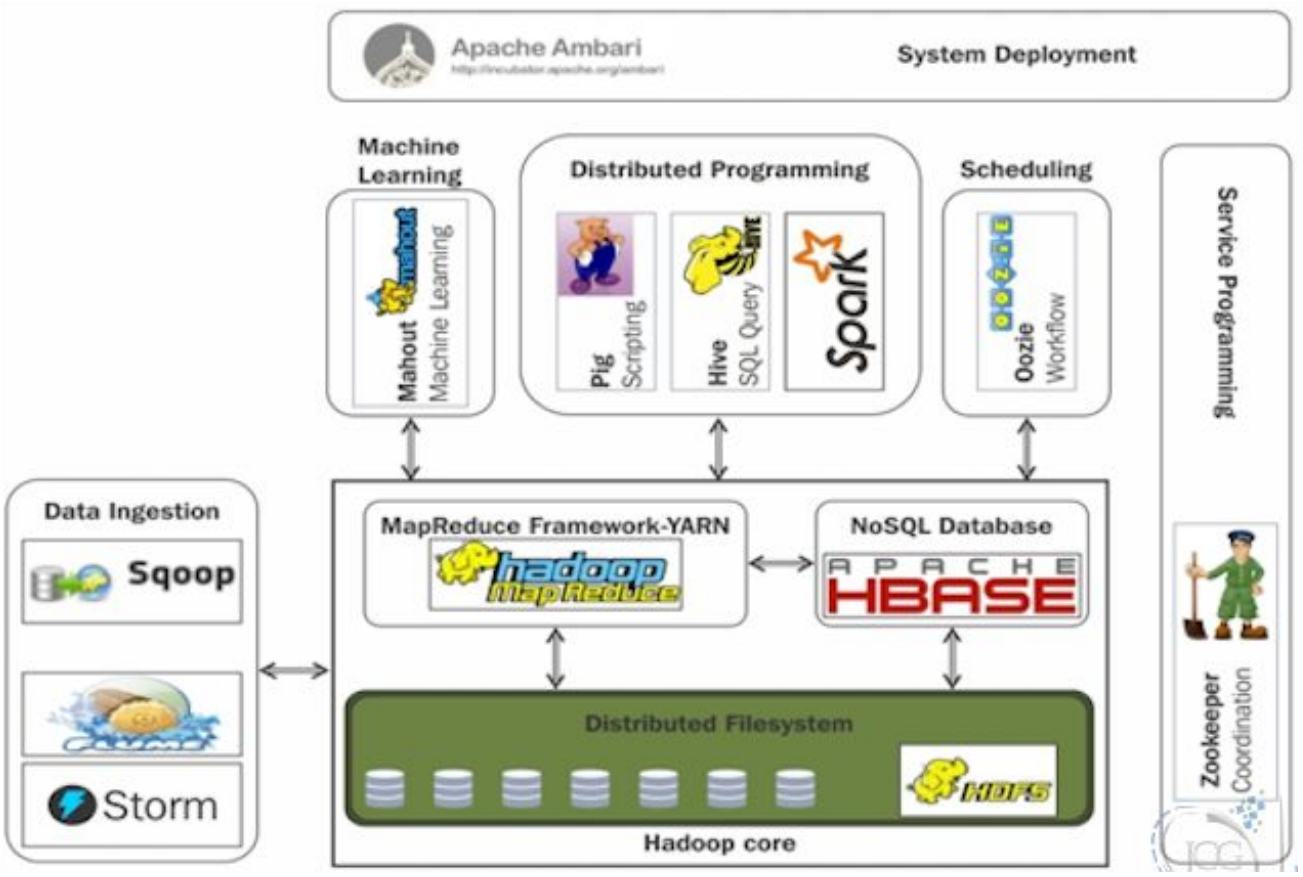
HADOOP ECOSYSTEM TOOLS

In addition to the core components of Hadoop mentioned above, there are many other data access tools that together form the 'Hadoop ecosystem' each serving a particular purpose. The table below describes some of the most commonly used tools.

Tools	***Purpose / Use Cases***
Apache Pig	A high-level data-flow scripting language (called *Pig Latin*) that can be used to write MapReduce jobs, without using Java: ● Code is translated to MapReduce programs without having to write MapReduce code ● Utilises MapReduce in a distributed system while simplifying its developing and executing jobs ● For *batch* processing & applications ● *Not* intended for real-time querying
Apache Hive	An SQL engine that can be used to query data in HDFS, using a language called *HiveQL* (similar syntax to SQL). To use this, HiveQL writes and processes high-level code that is first translated to MapReduce programs. HiveQL is great for: ● Periodic and ad-hoc queries of data in HDFS 'the SQL way' rather than via Java or scripting languages

	• Combining structured data residing in an external RDBMS (like Oracle) with data already residing in HDFS • Ideal for *batch* processing & applications • *Not* intended for real-time querying
Apache HBase	A Non-relational database (or a NoSQL database) that sits directly atop HDFS, and allows for real-time querying. Unlike Apache Hive and MapReduce, it is: • *Not intended* for batch processing & applications • *Ideal* for real-time querying HBase is fast because it makes use of in-memory processing engines that allow for faster read/write (or I/O) operations compared to Hive, Pig and MapReduce: • Sits directly above HDFS, so it doesn't need to first translate into MapReduce in order to query data • Makes it appropriate for interactive analytics (i.e., pulling and re-pulling data quickly for analysis in R or Python) • Appropriate for building dashboards quickly
Apache Mahout	Mahout provides the analytical toolset that directs Hadoop to analyse and yield analytical results. It also provides executable Java code to run algorithms for several techniques, and is scalable for Big Data: • Clustering (e.g., k-means) • Classification (e.g., Naïve Bayes, logistic regression, random forests) • Collaborative filtering / Recommenders
Apache Sqoop	An interface that facilitates the transfer of data between Hadoop and standard RDBMS (sitting outside the Hadoop ecosystem) and other structured data stores.
Apache Flume	A tool designed to collect, aggregate, and transfer *streaming data* into Hadoop, particularly to log and event data. • A key component of the architecture of streaming applications • Good for anomaly detection applications (e.g., fraud detection)
Apache Oozie	A workflow scheduling system that can be integrated with the rest of the stack to manage the workflow of Hadoop jobs.
Apache ZooKeeper	A database/cluster that centrally manages distributed applications (or 'zoos'). It offers services like maintaining configuration information and the naming registry (or registries). Instead of using its own coordination and configuration services, HBase uses ZooKeeper.

Below is a map of the Hadoop Ecosystem, and how all the tools fit together.



PRACTICE QUESTIONS:

1. Define “data warehousing”
2. Define “data repository”
3. Name and define two data storage tools in the Hadoop ecosystem
4. Name and define two data processing tools in the Hadoop ecosystem
5. Name and define three data access tools in the Hadoop ecosystem

NOSQL

NoSQL (“Not only SQL”) databases are *data stores* often used for *unstructured data*. Its main advantage is linearly scaling with increases in data. This is done by adding more machines to the distributed system. See the table below for a description of the different kinds of data stores that are used with NoSQL: ***Key/Value Stores, Document Stores, Column Family Stores,*** and ***Graph Databases.***

Type:	***Description:***	***Examples:***
Key/Value Stores	The 'Value' here is the data we need to access, and the 'Key' is the identifier we would use to access it. This structure allows for more complexity. ‘Value’ can be a set of key/value pairs, which themselves can branch out into more key/value pairs, etc. *Example:* Web traffic applications, where ‘key’ is the session ID, and ‘value’ is the data captured during the session.	Oracle NoSQL, Redis
Document Stores	Key/Value stores where the values are semi-structured data files like JSON or XML.	MongoDB, CouchDB, DocumentDB from AWS
Column Family Stores	Also a Key/Value store, but the key represents a family of columns (e.g., demographic features). Column family stores are useful when dealing with sparse data (i.e., many columns with few entries). One example is storing user feedback on products.	Apache HBase, Apache Cassandra, Apache Accumulo
Graph Databases	Used to store network graphs. They use nodes to store data entities (e.g., people, web page links, or items) and edges to store the relationships between those entities. While RDBMS can be used to store this data, it is much easier to navigate relationships between nodes in a graph database. This has clear use cases in social networks, like Facebook and LinkedIn, but also geospatial applications like delivery/traffic systems.	OrientDB, Neo4j

2. BASIC & ADVANCED SQL

OVERVIEW

This is a summary of some of the Advanced SQL topics, including window functions, grouping extensions, user-defined functions, MADlib, and in-database text analytics.

KEY CONCEPTS & TERMINOLOGY

Basic SQL clauses

Window Functions

System Functions

External Library Functions (MADlib)

Grouping Extensions

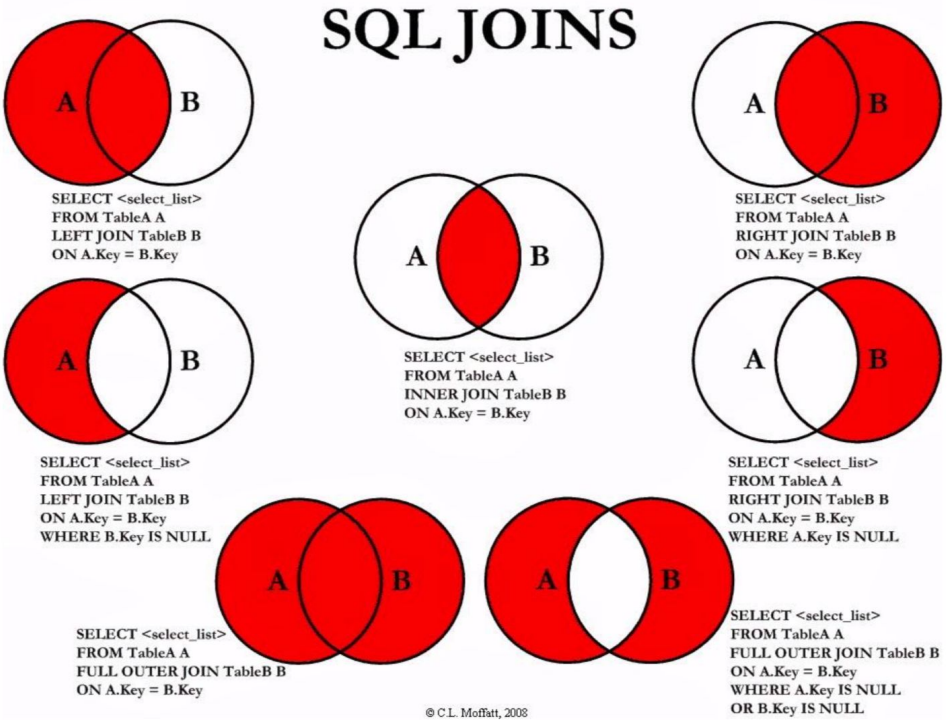
In-database Text Analytics

User-defined Functions

BASIC SQL CLAUSES

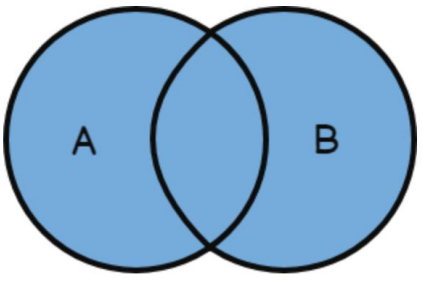
JOINS

Join clauses are used to combine rows from two or more tables by using a related column between the tables. There are five main types of JOIN clauses: INNER, LEFT OUTER, RIGHT OUTER, FULL OUTER and CROSS. As a special case, a table (base table, view, or joined table) can JOIN to itself in a self-join. Below is an example of how the different join clauses function to select rows of data from different tables. Additional join types can be seen here.



UNIONS

Union clauses are used to combine two separate SELECT statements, and produce the result set as the union (combination) of both the SELECT statements. The fields defined in both the SELECT statements must be in the same order, the same number, and the same data types.
The Union clause produces distinct values in the result set. To get duplicate values in a union, you must use **UNION ALL.**



GROUP BY clauses in SQL statements collect our rows into groups on the basis of certain columns.

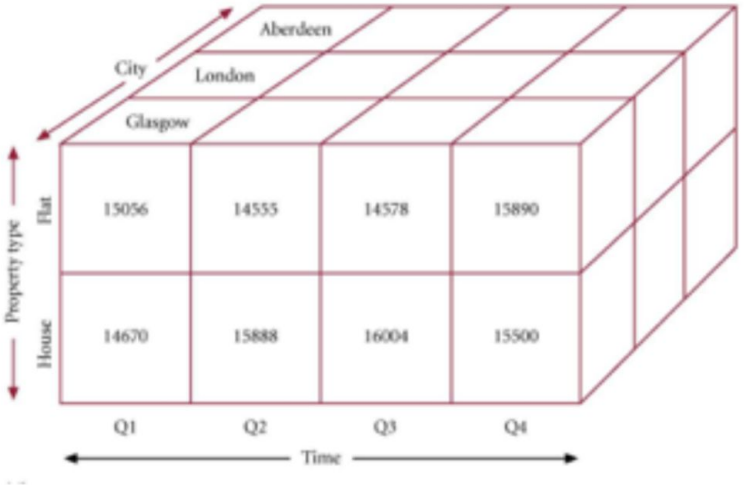
GROUPING EXTENSIONS

OLAP

Grouping extensions extend the kind of work done by the GROUP BY command. They often involve the use of OLAP (Online Analytic Processing) the dynamic synthesis, analysis, and consolidation of large volumes of multi-dimensional data.

OLAP provides support for complex calculations, sometimes involving multi-dimensional data (e.g., property sales *revenue*, and associations with property *location,* and *time* of property sale). We can represent such data in a standard table, or as a 2- or 3-d matrix (known as a **Data Cube**):

Property Type	City	Time	Total Revenue
Flat	Glasgow	Q1	15056
House	Glasgow	Q1	14670
Flat	Glasgow	Q2	14555
House	Glasgow	Q2	15888
Flat	Glasgow	Q3	14578
House	Glasgow	Q3	16004
Flat	Glasgow	Q4	15890
House	Glasgow	Q4	15500
Flat	London	Q1	19678
House	London	Q1	23877
Flat	London	Q2	19567
House	London	Q2	28677
.....	.....	.....	.....
.....	.....	.....	.....



Representing data as **Data Cubes** with OLAP can make the outputs of aggregates and ordered aggregates - such as COUNT() and SUM() functions - more easy to visualise.

The **Grouping Extensions** are the functions we can use to facilitate these aggregates on multi-dimensional data. The Grouping Extensions include the ROLLUP and CUBE functions, which we will cover individually.

ROLLUP enables a SELECT statement to calculate multiple levels of subtotals across a specified group of dimensions.

ROLLUP: EXAMPLE

Suppose we work with an estate agency's database that contains at least the following tables: Branch (for the agency's branches), PropertyForSale (no surprises) and PropertySale (for all sold properties).

We want to show the totals for sales of flats or houses by branch offices located in Aberdeen, Edinburgh or Glasgow for the months of August and September 2013. ROLLUP can divide this into two steps:

Step 1: we get a level of subtotals which aggregate across cities for each combination of propertyType and yearMonth.

Step 2: we aggregate across yearMonth and city for each propertyType. ROLLUP gives us the grand total.

Look at the query below using ROLLUP. If we wrote this, we would get the table below:

```
SELECT propertyType, yearMonth, city,
SUM(saleAmount) AS sales
FROM Branch, PropertyForSale, PropertySale
WHERE Branch.branchNo = PropertySale.branchNo
AND PropertyForSale.propertyNo =
PropertySale.propertyNo
AND PropertySale.yearMonth IN ('2013-08',
'2013-09')
AND Branch.city IN ('Aberdeen', 'Edinburgh',
'Glasgow')
GROUP BY ROLLUP(propertyType, yearMonth, city);
```

propertyType	yearMonth	city	sales
flat	2013-08	Aberdeen	115422
flat	2013-08	Edinburgh	236571
flat	2013-08	Glasgow	7664
flat	2013-08		359657
flat	2013-09	Aberdeen	121700
flat	2013-09	Edinburgh	312400
flat	2013-09	Glasgow	8755
flat	2013-09		442855
flat			802512
house	2013-08	Aberdeen	777987
house	2013-08	Edinburgh	543670
house	2013-08	Glasgow	42763
house	2013-08		1364420
house	2013-09	Aberdeen	726321
house	2013-09	Edinburgh	1665033
house	2013-09	Glasgow	43889
house	2013-09		2435243
house			3799663

CUBE

CUBE clause takes a specified set of grouping columns, and creates subtotals for all possible combinations that can be calculated for a Data Cube with specified dimensions. CUBE is typically used in queries using columns from multiple dimensions, rather than columns representing different levels of a single dimension.

CUBE generally appears in the GROUP BY clause, and can be used in any situation that requires cross-tabular reports (e.g., reports in the SQL online practice). Even with a few columns, CUBE statements can produce very long tables. This is why representing data with Data Cubes can sometimes be preferable to tables.

Generally, the data we need can be generated with one CUBE statement. As an example, we are at an estate agency, and we want to show all possible subtotals for sales of properties by branch office in Aberdeen, Edinburgh, and Glasgow for August and September 2013.

If we wrote the following code, we get a much longer table (image right) than we would have if we only used ROLLUP:

```
SELECT propertyType, yearMonth, city, SUM(saleAmount)
AS sales
FROM Branch, PropertyForSale, PropertySale
WHERE Branch.branchNo = PropertySale.branchNo
AND PropertyForSale.propertyNo =
PropertySale.propertyNo
AND PropertySale.yearMonth IN ('2013-08', '2013-09')
AND Branch.city IN ('Aberdeen', 'Edinburgh', 'Glasgow')
GROUP BY CUBE(propertyType, yearMonth, city);
```

propertyType	yearMonth	city	sales
flat	2013-08	Aberdeen	115422
flat	2013-08	Edinburgh	236571
flat	2013-08	Glasgow	7664
flat	2013-08		359657
flat	2013-09	Aberdeen	121700
flat	2013-09	Edinburgh	312400
flat	2013-09	Glasgow	8755
flat	2013-09		442855
flat		Aberdeen	237122
flat		Edinburgh	548971
flat		Glasgow	16419
flat			802512
house	2013-08	Aberdeen	777987
house	2013-08	Edinburgh	543670
house	2013-08	Glasgow	42763
house	2013-08		1364420
house	2013-09	Aberdeen	726321
house	2013-09	Edinburgh	1665033
house	2013-09	Glasgow	43889
house	2013-09		2435243
house		Aberdeen	1504308
house		Edinburgh	2208703
house		Glasgow	86652
house			3799663
	2013-08	Aberdeen	893409
	2013-08	Edinburgh	780241
	2013-08	Glasgow	50427
	2013-08		1724077
	2013-09	Aberdeen	848021
	2013-09	Edinburgh	1977433
	2013-09	Glasgow	52644
	2013-09		2878098
		Aberdeen	1741430
		Edinburgh	2757674
		Glasgow	103071
Total			4602175

IN-DATABASE TEXT ANALYSIS

REGULAR EXPRESSIONS

While SQL databases are typically associated with structured data, databases often contain unstructured text data as well (e.g. comments, descriptions, freeform text). You can use REGULAR EXPRESSIONS to select a subset of the rows within a table. If you recall from Text Analysis, a regular expression ("regex") is a sequence of characters defining a search pattern used to find parts of strings. See the table below for regex examples that can be used in SQL:

Element	Description	Example	Output	Explanation
~	Contains the regular expression (case sensitive)	SELECT '123a567' ~ '23|b'	TRUE	The string '23' occurs in '123a567'
|	Matches item a or b (a | b)	SELECT '123a567' ~ '32|b'	FALSE	Neither '32' nor 'b' occur in '123a567'
^	Looks for matches at the beginning of a string	SELECT '123a567' ~ '^123a'	TRUE	123a567' begins with '123a'
$	Looks for matches at the end of the string	SELECT '123a567' ~ '27$'	FALSE	The string doesn't end with '27'
.	Matches any single character	SELECT '123a567' ~ '2..5'	TRUE	There are exactly two characters between '2' and '5' in the string '123a567'
*	Matches preceding item zero or more times	SELECT '123a567' ~ '37*'	TRUE	The * 'binds' only to the character '7'. So what '37*' is asking is whether zero or more 7s follows the occurrence of a 3 in the string '123a567', and the answer is yes.
+	Matches preceding item one or more times	SELECT '123a567' ~ '2+a'	FALSE	It's not the case that there's one or more 2s, followed directly by an a, in the string '123a567'
?	Makes the preceding item optional	SELECT '123a567' ~ '7?a'	TRUE	This effectively asks 'is there an 'a' in the string '123a567'? And the answer is yes
{n}	Matches the preceding item exactly n times	SELECT '1235567' ~ '5{2}'	TRUE	5 does occur twice in the string
( )	Matches the contents exactly	SELECT '123a567' ~ '(23a5)7*'	TRUE	23a5 occurs in sequence, then 7 occurs 0 times; consistent with * applying to it
[ ]	Matches any of the characters in the content, such as [0-9]	SELECT '123a567' ~ [23 a5]	TRUE	The substring occurs in the string
\\x	Matches a nonalphanumeric character named x	SELECT '$50K+' ~ '\\$'	TRUE	The nonalphanumeric symbol '$' occurs in the string
\\y	Matches an escape string \y	SELECT '123a567' ~ '\\w'	TRUE	Provided that \\w stands for the characters 0-0 or a-z

WINDOW FUNCTIONS

Window functions enable a level of subtotal analysis similar to the grouping extensions ROLLUP and CUBE, but allow you to maintain granular row data. They are also able to access more than just the current row of a query result. Windows functions do not modify columns, but create new columns with the desired output.

There are a couple of things to keep in mind when using Window functions:

- They are *always* associated with the SQL clause *OVER* (i.e., window functions can perform aggregations without using GROUP BY)
- You can use multiple window functions to add multiple columns with subtotal analyses

In the following example, we select customers, orders, and suborders from a database. We would like to display customer and order subtotals alongside each suborder. With ROLLUP we could generate summaries but we would lose the individual rows. However, we can use Window functions to accomplish this:

Example using basic SQL joins:

```
SELECT c.customerName,
       ord.orderNumber,
       od.quantityOrdered,
       od.priceEach,
       (od.quantityOrdered * od.priceEach) as orderSubTotal
FROM customers c
       LEFT JOIN orders ord ON ord.customerNumber = c.customerNumber
       LEFT JOIN orderdetails od ON od.orderNumber = ord.orderNumber
```

customerName	orderNumber	quantityOrdered	priceEach	orderSubTotal
Alpha Cognac	10178	34	86.90	2954.60
Alpha Cognac	10178	30	64.15	1924.50
Alpha Cognac	10178	21	68.77	1444.17
Alpha Cognac	10178	27	65.75	1775.25
Alpha Cognac	10178	34	67.82	2305.88
Alpha Cognac	10178	48	104.81	5030.88
Alpha Cognac	10178	41	70.54	2892.14
Alpha Cognac	10178	42	127.73	5364.66
Alpha Cognac	10178	24	131.92	3166.08
Alpha Cognac	10136	41	169.34	6942.94
Alpha Cognac	10136	36	120.91	4352.76
Alpha Cognac	10136	25	117.48	2937.00
Alpha Cognac	10397	34	52.96	1800.64
Alpha Cognac	10397	36	80.44	2895.84
Alpha Cognac	10397	48	86.15	4135.20
Alpha Cognac	10397	22	62.88	1383.36
Alpha Cognac	10397	32	69.29	2217.28
Alpha Cognac	10178	45	41.71	1876.95

Example using Window functions:

```
SELECT c.customerName,
       ord.orderNumber,
       od.quantityOrdered,
       od.priceEach,
       (od.quantityOrdered * od.priceEach) as orderSubTotal,
SUM(od.quantityOrdered * od.priceEach) OVER(PARTITION BY ord.orderNumber) as orderTotal,
SUM(od.quantityOrdered * od.priceEach) OVER(PARTITION BY c.customerNumber) as customerTotal
FROM customers c
       LEFT JOIN orders ord ON ord.customerNumber = c.customerNumber
       LEFT JOIN orderdetails od ON od.orderNumber = ord.orderNumber
```

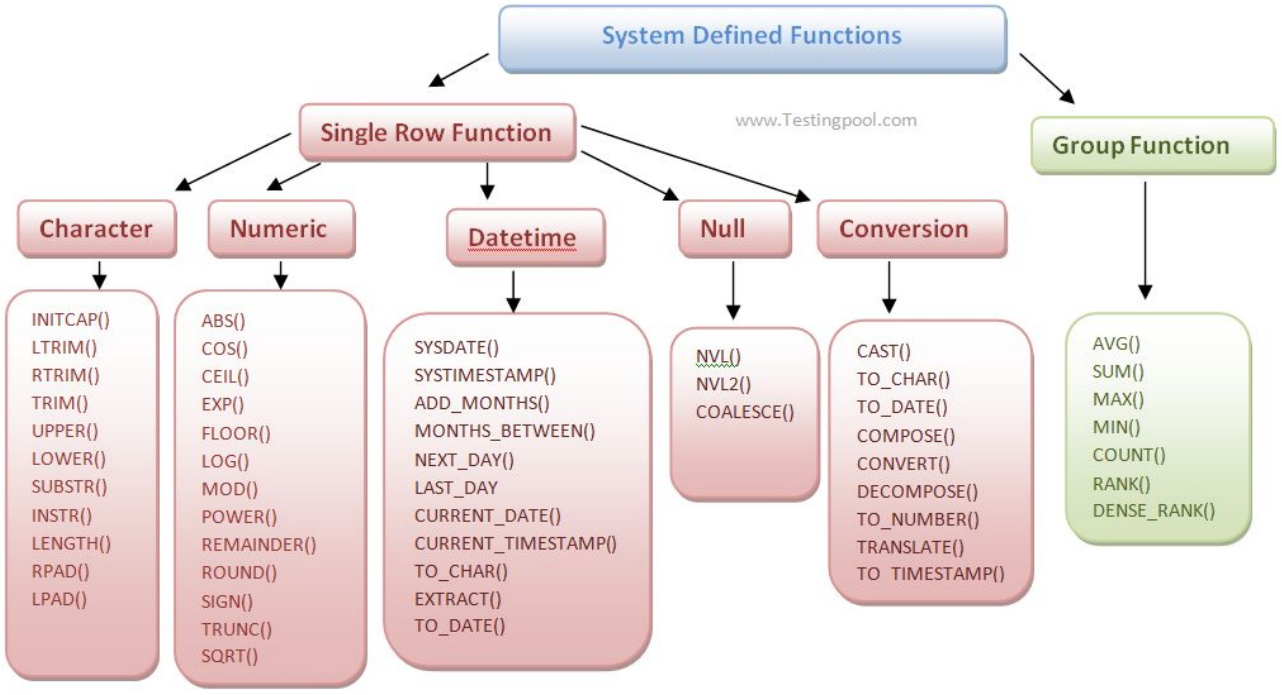
customerName	orderNumber	quantityOrdered	priceEach	orderSubTotal	orderTotal	customerTotal
Alpha Cognac	10136	25	117.48	2937.00	14232.70	60483.36
Alpha Cognac	10136	36	120.91	4352.76	14232.70	60483.36
Alpha Cognac	10136	41	169.34	6942.94	14232.70	60483.36
Alpha Cognac	10178	24	131.92	3166.08	33818.34	60483.36
Alpha Cognac	10178	42	127.73	5364.66	33818.34	60483.36
Alpha Cognac	10178	41	70.54	2892.14	33818.34	60483.36
Alpha Cognac	10178	48	104.81	5030.88	33818.34	60483.36
Alpha Cognac	10178	34	67.82	2305.88	33818.34	60483.36
Alpha Cognac	10178	27	65.75	1775.25	33818.34	60483.36
Alpha Cognac	10178	21	68.77	1444.17	33818.34	60483.36
Alpha Cognac	10178	30	64.15	1924.50	33818.34	60483.36
Alpha Cognac	10178	34	86.90	2954.60	33818.34	60483.36
Alpha Cognac	10178	22	91.74	2018.28	33818.34	60483.36
Alpha Cognac	10178	45	68.11	3064.95	33818.34	60483.36
Alpha Cognac	10178	45	41.71	1876.95	33818.34	60483.36
Alpha Cognac	10397	32	69.29	2217.28	12432.32	60483.36
Alpha Cognac	10397	22	62.88	1383.36	12432.32	60483.36
Alpha Cognac	10397	48	86.15	4135.20	12432.32	60483.36

There are many additional Window functions that can be used in SQL:

Function	Return Type	Description
`row_number()`	**bigint**	number of the current row within its partition, counting from 1
`rank()`	**bigint**	rank of the current row with gaps; same as `row_number` of its first peer
`dense_rank()`	**bigint**	rank of the current row without gaps; this function counts peer groups
`percent_rank()`	**double precision**	relative rank of the current row: (`rank` - 1) / (total rows - 1)
`cume_dist()`	**double precision**	relative rank of the current row: (number of rows preceding or peer with current row) / (total rows)
`ntile(`***num_buckets*** `integer)`	**integer**	integer ranging from 1 to the argument value, dividing the partition as equally as possible
`lag(`***value*** `anyelement [,` ***offset*** `integer [,` ***default*** `anyelement ])])`	**same type as *value***	returns ***value*** evaluated at the row that is ***offset*** rows before the current row within the partition; if there is no such row, instead return ***default*** (which must be of the same type as ***value***). Both ***offset*** and ***default*** are evaluated with respect to the current row. If omitted, ***offset*** defaults to 1 and ***default*** to null
`lead(`***value*** `anyelement [,` ***offset*** `integer [,` ***default*** `anyelement ])])`	**same type as *value***	returns ***value*** evaluated at the row that is ***offset*** rows after the current row within the partition; if there is no such row, instead return ***default*** (which must be of the same type as ***value***). Both ***offset*** and ***default*** are evaluated with respect to the current row. If omitted, ***offset*** defaults to 1 and ***default*** to null
`first_value(`***value*** `any)`	**same type as *value***	returns ***value*** evaluated at the row that is the first row of the window frame
`last_value(`***value*** `any)`	**same type as *value***	returns ***value*** evaluated at the row that is the last row of the window frame
`nth_value(`***value*** `any,` ***nth*** `integer)`	**same type as *value***	returns ***value*** evaluated at the row that is the ***nth*** row of the window frame (counting from 1); null if no such row

SYSTEM FUNCTIONS

System functions in SQL are ones that are built into the SQL system. There is a variety of system functions that handle both single-row and group functions, as seen in the below image:



USER DEFINED FUNCTIONS

If you have a need that is not met by a built-in SQL function, it is possible to create your own:

- The keyword required at the beginning is `CREATE` (e.g. `CREATE FUNCTION`)
- Create different types of functions, (e.g., aggregate function: `CREATE AGGREGATE`)

EXTERNAL LIBRARY FUNCTIONS (MADlib)

As well as user-defined functions, it is possible to use external library functions like MADlib. MADlib is an open source library designed to allow scalable in-database analytics. MADlib makes it possible to perform different methods, including:

- Classification
- Regression
- Clustering
- Topic modelling
- Association rule mining
- Descriptive statistics
- Validation

PRACTICE QUESTIONS:

1. What are System Functions? List 4 examples.

2. What are Window Functions? List 4 examples.

3. What are User-defined Functions?

OPERATIONALISING AN ANALYTICS PROJECT & DATA VISUALISATION TECHNIQUES

VISUALISATION

OVERVIEW

This is a one-page summary about using different visualisations in various contexts. This includes the different characteristics of visuals that are appropriate for data analysis purposes vs. communication purposes; and types of graphs for data (categorical, continuous, combined) and the corresponding R code to make them.

KEY CONCEPTS & TERMINOLOGY

Communicating an Analytics Project	Presentations Requirements
Data-Ink Ratio	Graphs: Continuous Data
Graphs: Categorical Data	Graphs: Combined Data

COMMUNICATING AN ANALYTICS PROJECT

Communication of your findings is part of the final phase in the data lifecycle. During this phase, you will need to clearly present the project outcome and benefits to other team members, stakeholders, managers, and others. These outputs can be grouped into 4 main deliverables:

4 Main Deliverables	
1. Presentations for Project Sponsors: Should be *high-level takeaways* for executive *stakeholders*, contain only a *few key messages* to aid the decision-making process. Should contain *clean, easy visuals*.	*2. Presentations for Analysts:* Describes the changes to *business processes* and reports to *data scientists* (or similar). Use *technical graphs* (ROC curve, box plots), as they will be interested in analytical details.
3. Code: For technical people managing the production environment (e.g., engineers)	*4. Technical specifications:* For implementing code

PRESENTATIONS REQUIREMENTS: PROJECT SPONSORS VS. ANALYSTS

As some of the information will be the same for multiple audiences, it is useful to create some core materials that can be used in presentations to both executive/sponsor and technical audiences alike. For the most part, there are 7 core components for any presentation, but certain components - approach, key points, model details, and recommendations - need to be tailored based upon who is in the main audience.

FEATURES OF PRESENTATIONS FOR ANALYSTS VS. PROJECT SPONSORS

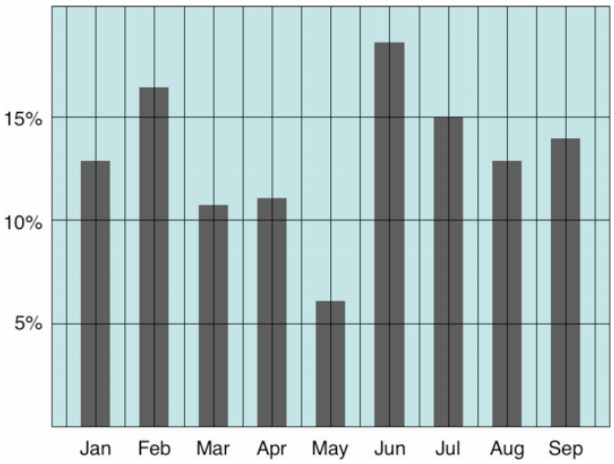
Analysts & Technical Audiences	***Project Sponsors & Executives***
Components: 1. *Project goals*: top 3-5 goals 2. *Main findings*: emphasise key messages 3. *Approach*: high-level & relevant details 4. *Model description*: brief overview 5. *Key points*: analysis-oriented graphs; visuals of key variables and significance 6. *Model details*: discuss model types, key variables, code/technology used. 7. *Recommendations*: model implications or deployment to production environment	*Components:* 1. *Project goals*: top 3-5 goals 2. *Main findings*: emphasise key messages 3. *Approach*: high-level 4. *Model description*: brief overview 5. *Key points*: simple charts/graphs (e.g., bar charts, line graphs) 6. *Model details*: omit (or very high-level) 7. *Recommendations*: business impact (e.g., risks, ROI), and key points to help use this information within the organisation
Graphical features: Graphs should convey more detailed and/or technical information about the data. This could be used for data exploration, hypothesis formation, data analysis, and/or data interpretation.	*Graphical features:* Graphs should be clean, clear, and simple in both layout and the amount of data shown. Avoid complicated or technical graphs (3D graphs, box plots) and stick to simpler charts. Use emphasis colour sparingly.
Graphs to use: ● Scatterplots ● Heatmaps ● ROC curves ● Box plots ● Histograms ● Density curves ● Hexbin graphs ● Decision trees	*Graphs to use:* ● Bar charts (incl. stacked & grouped) ● Line graphs ● Bubble charts ● Gantt charts ● Heatmaps ● Pie charts** ● Spider / radar charts** ***Use sparingly and with extreme caution*

DATA-INK RATIO

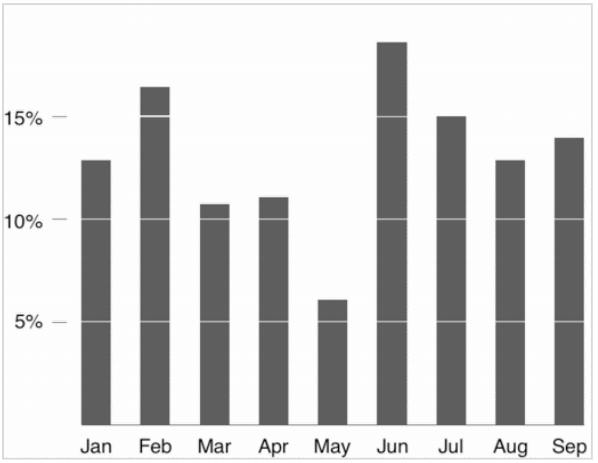
The data-ink ratio is a concept that is widely used for creating effective visualisations. It is the proportion of ink used to present actual data (e.g., the actual graphs or charts) compared to the total amount of ink used in the entire display (e.g., the graphs & titles, legends, axis gridlines, tick marks, and other information not directly needed or relevant to understanding the data).

Good visualisations should have a high ratio of data-ink to non-data-ink. The two graphs below are examples of graphs with a low data-ink ratio (i.e., a lot of excess ink not necessary to understand the data), and one with a high data-ink ratio (i.e., lots of ink relating to the data, with little unnecessary ink unrelated to the data). You should aim to create graphs that use as little excess ink as possible on graphical features that do not directly help to understanding a graph. See the example images below for low- and high- data-ink ratios.

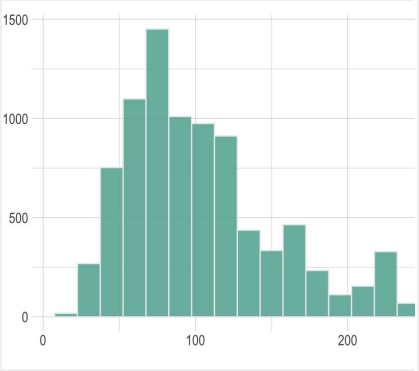
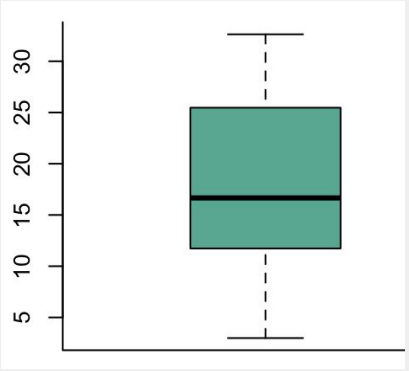
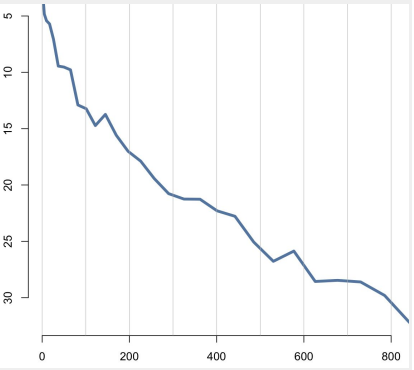
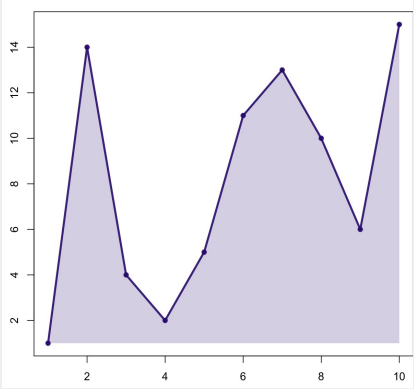
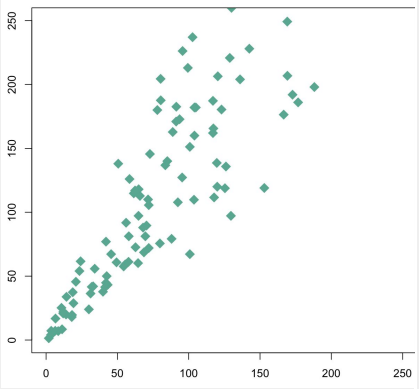
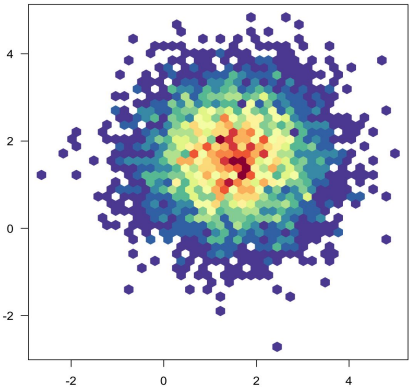
Low data-ink ratio:



High data-ink ratio:



GRAPHS: CONTINUOUS (WITH R CODE)

Histogram	***Box Plot***	***Line Graph***
		
R code: hist(df$x)	*R code:* boxplot(x=, data=df)	*R code:* lines(dfx, dfy)
Area Plot	***Scatter Plot***	***Hexbin Plot***
		
R code (in ggplot2): ggplot(df, aes(x=, y=,)) + geom_area()	*R code:* plot(dfx, dfy)	*R code (in ggplot2):* ggplot(df, aes(x=, y=)) + geom_hex()

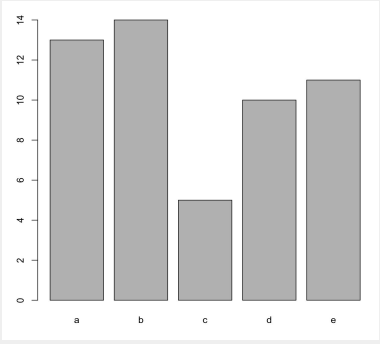
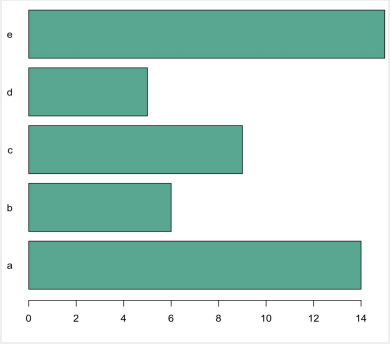
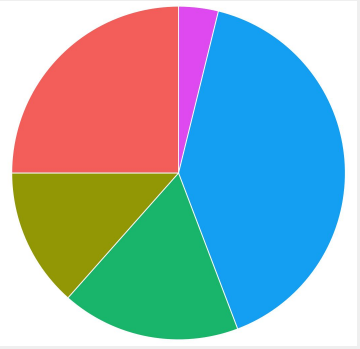
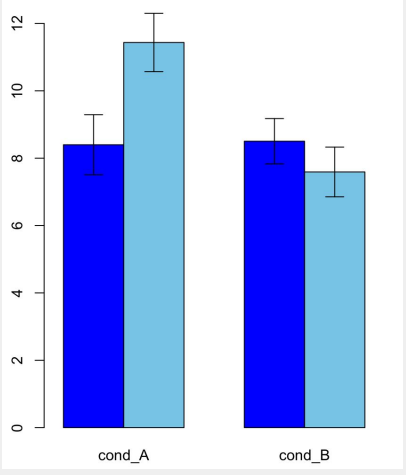
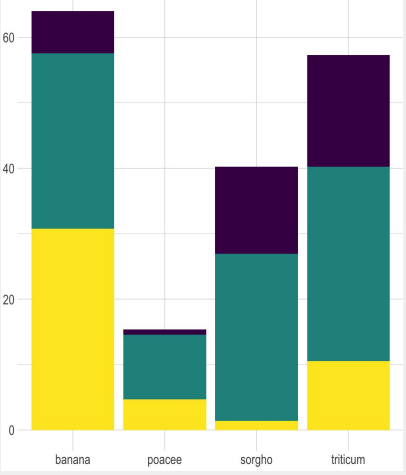
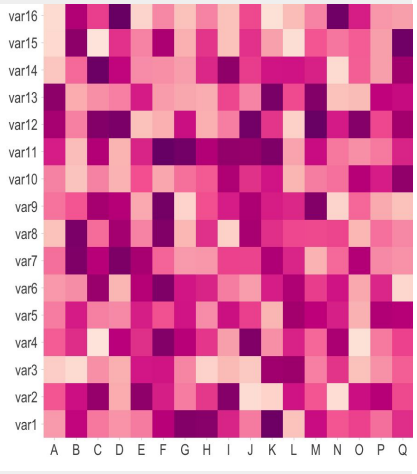
PRACTICE QUESTIONS:

1. Describe the 4 main deliverables from an analytics project.

2. What are the main differences in presentations for project sponsors versus technicians/analysts?

3. What are the key characteristics of visualisations for project sponsors versus technicians/analysts?

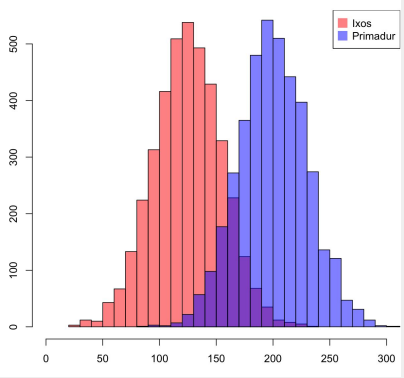
GRAPHS: CATEGORICAL (WITH R CODE)

Bar Chart (vertical)	***Bar Chart (horizontal)***	***Pie Chart****
		
R code: count <- table(df$x1) barplot(count)	*R code:* count <- table(df$x1) barplot(count, horiz=TRUE)	*R code:* pie(slice_values, slice_labels)
Bar Chart (grouped)	***Bar Chart (stacked)***	***Heat Map***
		
R code: count <-table(df$x1, df$x2) barplot(count, beside=TRUE)	*R code:* count <-table(df$x1, df$x2) barplot(count)	*R code:* heatmap(df)

**Use with caution: there is still debate about whether these are useful graphs, and it is cautioned to avoid when possible

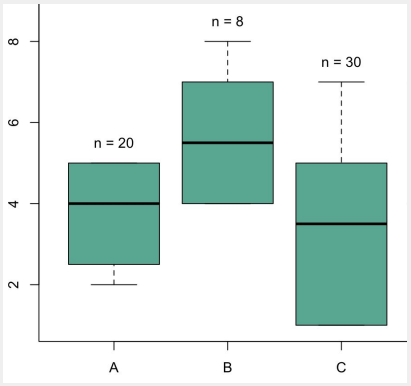
GRAPHS: COMBINED CONTINUOUS & CATEGORICAL (WITH R CODE)

Histogram (Grouped)



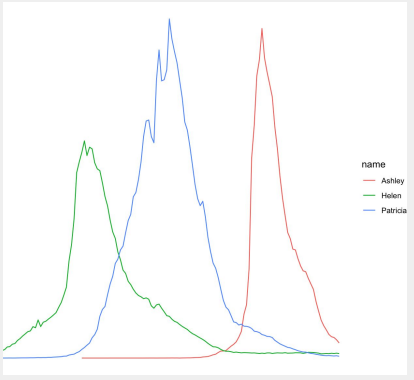
R code (in ggplot2):
ggplot(df, aes(x=, fill=)) +
geom_histogram( )

Box Plot (Groups)



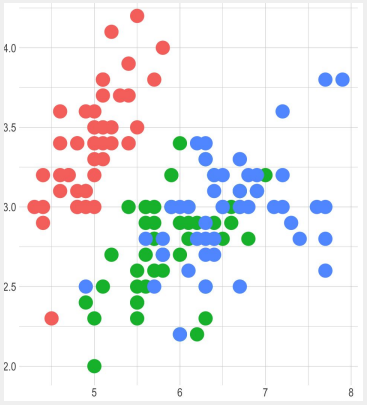
R code:
boxplot(y ~ x, data=df)

Line Graph (Groups)



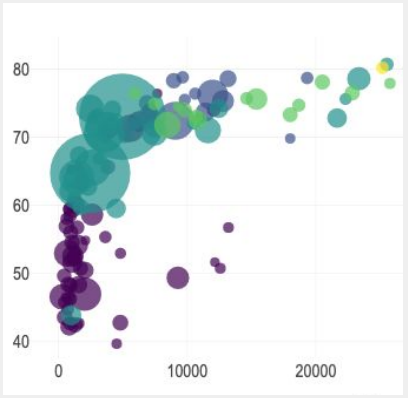
R code (in ggplot2):
ggplot(df, aes(x=, y=, group=))+
geom_line()

Scatter Plot (Grouped)



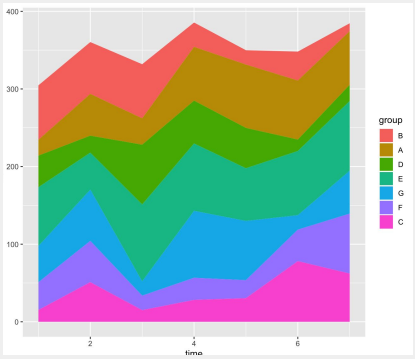
R code:
scatterplot(y ~ x1 | x2, data=df)

Bubble Chart



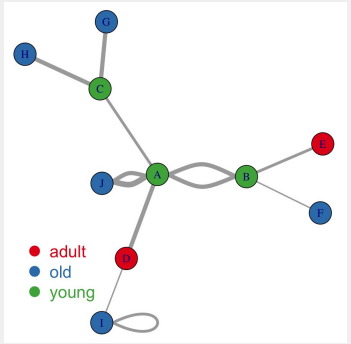
R code (in ggplot2):
ggplot(df, aes(x=, y=, size=)) +
geom_point()

Area Plot (Stacked groups)



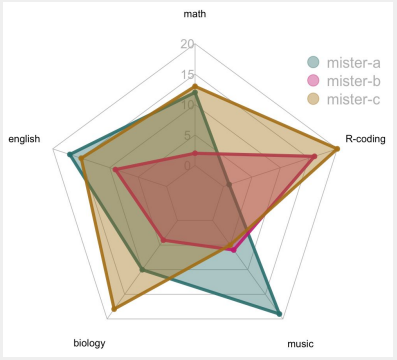
R code (in ggplot2):
ggplot(df, aes(x=, y=, fill=)) +
geom_area()

Network Graphs



R code:
net <- df(d=links, vertices=nodes, directed=F)
plot(*network*)

*Spider (Radar) Chart***



R code:
radarchart()

**Use with caution: there is still debate about whether these are useful graphs, and it is cautioned to avoid when possible

SELF-ASSESSMENT GUIDE

OVERVIEW

Use the following table to assess your development level for each topic. This way you can focus on the areas where you are at a lower level of 1 or 2 at first before progressing to review the topics where you are at a higher development level.

DEVELOPMENT LEVELS

1. **Very Uncomfortable / Very Low Confidence:**
   I don't recognise this topic or concept at all or have very little knowledge of it

2. **Uncomfortable / Low Confidence:**
   I can define and describe this concept but have low confidence in explaining, evaluating or applying it in more depth

3. **Comfortable / Moderate Confidence:**
   I can define, describe, explain or evaluate this concept at a moderate level and discuss its application

4. **Very Comfortable / High Confidence:**
   I can define, describe, explain, evaluate and apply this concept at a deep level and would feel confident in coaching someone else in this subject

CHECKLIST

TOPICS	1 - V. low	2 - Low	3 - Okay	4 - Great
BIG DATA, ANALYTICS & DATA SCIENTIST ROLE				
Big Data				
Characteristics of Big Data: The 4 V's				
Sources of Big Data				
Data Ecosystem				
Introduction to Data				
Data Types				
Data Scientist Role				
Business Drivers of Data				
DATA ANALYTICS LIFECYCLE				
Introduction to Data				
Data Analytics Lifecycle				
6 Phases of the Data Analytics Lifecycle				
INITIAL ANALYSIS OF THE DATA				

Introduction to Statistics				
Sample & Population Distributions				
z-tests				
Student's t-test				
Welch's t-test				
Wilcoxon Rank Sum Test				
ANOVA tests				
Statistical Equations				
ADVANCED ANALYTICS: THEORY & METHODS				
Clustering				
Clustering				
k-Means				
Centroid				
Within Sum of Squares				
Elbow Curve				
Clustering in Multi-dimensions				
Association Rules				
Sparse Matrix				
Transactions				
Itemsets				
Apriori Algorithm				
Support				
Lift				
Confidence				
Leverage				
Conviction				
Linear Regression				
Coefficient of Determination (r^2)				
Residuals				
Mean Absolute Error (MAE)				
Mean Square Error (MSE)				
Root Mean Square Error (RMSE)				
Logistic Regression				

Binary Logistic Regression				
Other Types of Logistic Regression				
Probability Threshold				
Naive Bayes Classifiers				
Bayes' Theorem				
Classifiers & Distributions				
Laplace's Correction (Smoothing)				
Decision Trees				
Decision Trees				
Information Gain				
Gini Index				
Confusion Matrix & Metrics				
ROC Curve				
AUC				
Time Series Analysis				
ARIMA Model				
Autoregressive Model (AR)				
Integrated Model (I)				
Moving Average Model (MA)				
Partial Autocorrelation Function (PACF)				
Stationarity & Differencing				
Autocorrelation Function (ACF)				
Box-Jenkins Method				
Akaike Information Criterion (AIC)				
Additive/Multiplicative Model Decomposition				
Text Analysis				
Natural Language Processing (NLP)				
Stop words (general and domain-specific)				
Regular expressions				
Tokenisation				
Stemming				
Lemmatisation				

Bag-of-words				
Sentiment analysis				
Topic extraction				
TF-IDF				
ADVANCED ANALYTICS: TECHNOLOGY & TOOLS				
Data Warehousing				
Data Repository				
Data Warehouse				
MapReduce				
YARN				
Hadoop Ecosystem				
NoSQL				
Basic & Advanced SQL				
Basic SQL clauses (joins, unions, group by)				
Grouping extensions (olap, rollup, cube)				
In-database text analytics				
System Functions				
User-Defined Functions				
External Library Functions (MADlib)				
OPERATIONALISING AN ANALYTICS PROJECT				
Visualisation				
Communicating an Analytics Project				
Presentation Requirements				
Data-Ink Ratio				
Graphs: Continuous Data				
Graphs: Categorical Data				
Graphs: Combined Data				