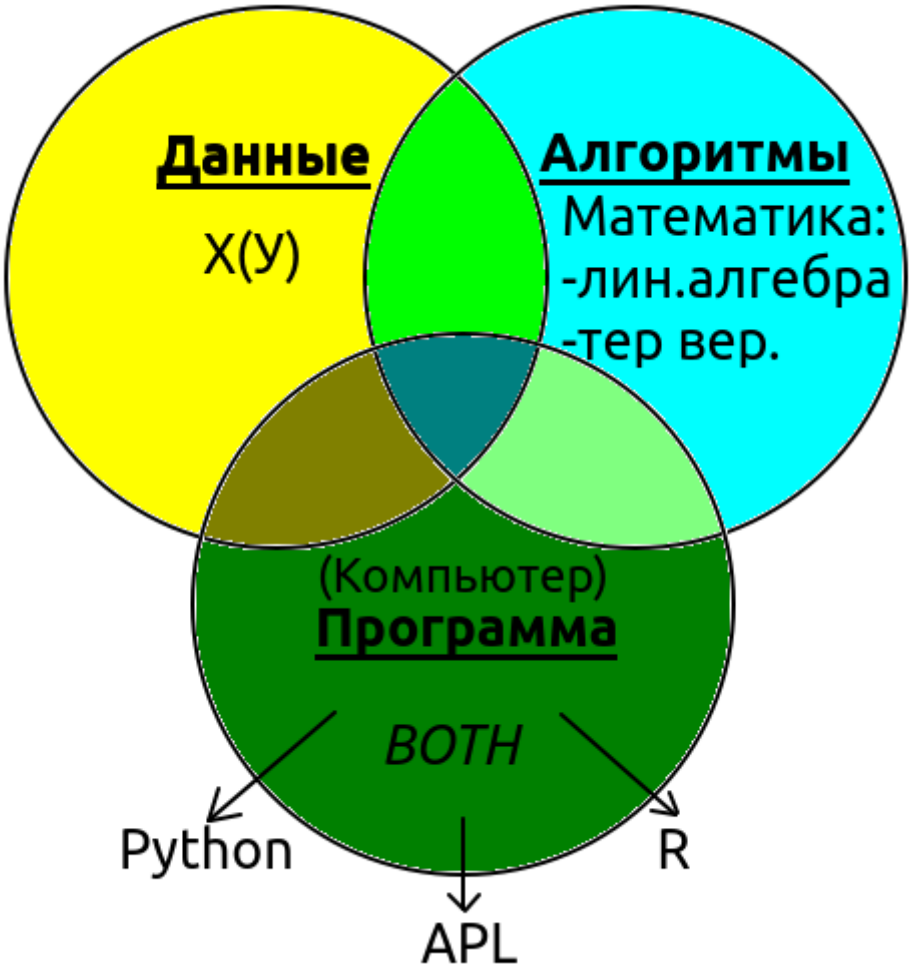
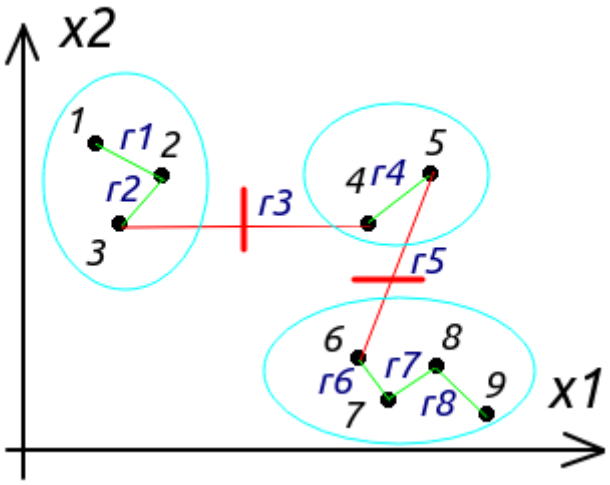
# Анализ данных (Data Science)



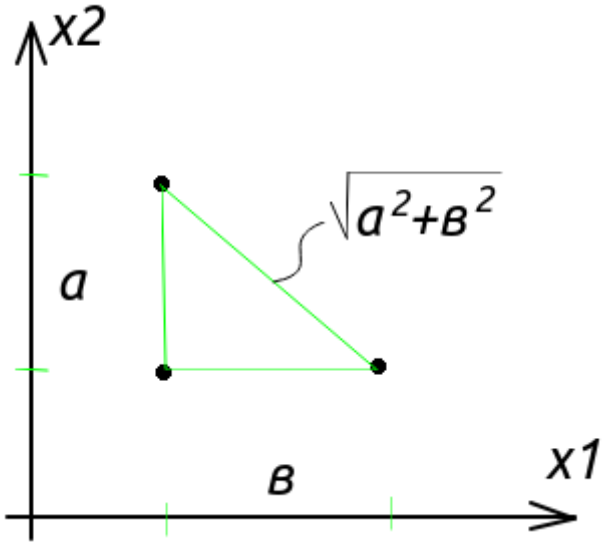

1. ***Данные*** - взяли пачку мед карт каких-то больных и списали из этой пачки лейкоциты в крови, сахар в моче, верхнее/нижнее давление, пульс и т.п., получился набор данных. Теперь с этими данными нужно что-то делать.
2. ***Математика*** - в этой математике нет каких-то теорем, лемм и пр., нельзя ничего заранее доказать не попробовав. Хотя под математикой в частности мы помечаем: лин.алгебра(техника работы с матрицами и векторами); теория вероятности(без всяких закидонов, достаточно понимать, какова частота наблюдать некое отклонение от среднего и т.д.) и прочие базовые направления.

   Выше матиматики следует упомянуть алгоритмы. Алгоритм - это последовательность действий, возможно содержащие некие условия, которая должна быть выполнена, чтобы получить результат который нас интересует. А результатом будет при отсутствии Y в наших данных - кластерный анализ.

   Важно! Что алгоритмы часто являются ЭВРИСТИЧЕСКИМИ, т.е. за ними не стоит математика каких-то теорем и т.д.. Например, какой-то Пупкин подумал: "Что если я для выделения групп буду использовать такую процедуру"



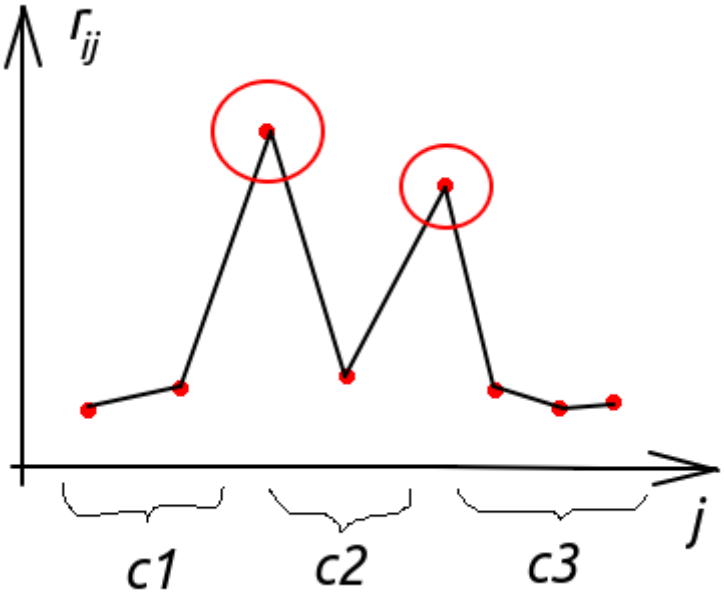

Пупкин захотел вычислить цепные расстояния. Расстояние между двумя точками на плоскости - это разность одноименных координат:




Расстояние в многомерном пространстве ничем не отличается:

$$r = sqrt(a^2 + b^2 + c^2 \ldots\ldots)$$

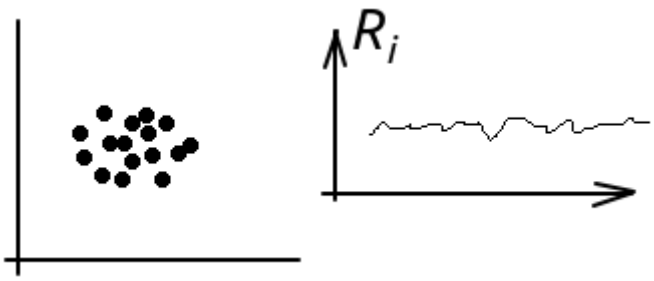
График цепных расстояний это значит, расстояние между первой и второй, между второй и третьей и т.д. точками.



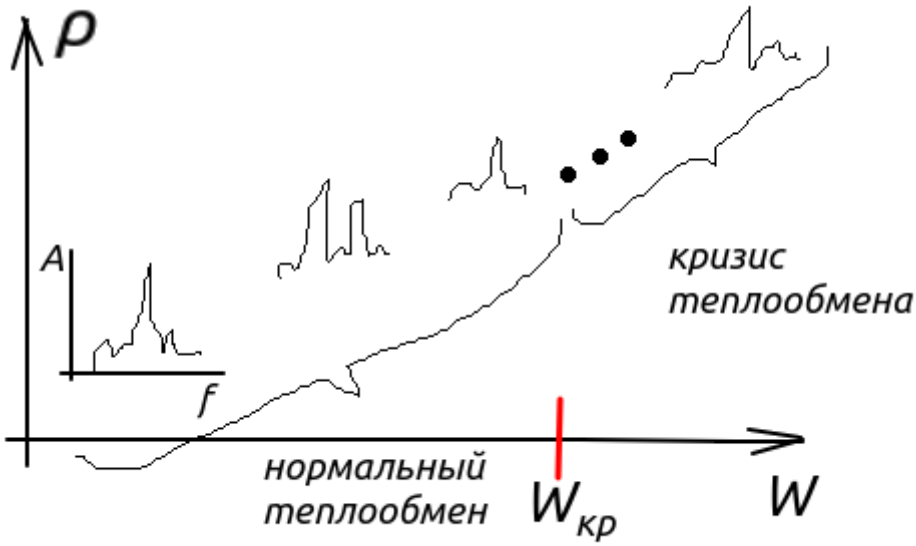
Максимум получился между третьей и четвертой, пятой и шестой точками.

Выделям кластеры: C1, C2, C3

Но пики могут быть не всегда, расстояние будет болтаться на каком-то одном уровне.




Этот алгоритм прост, но очень ущербен тем, что в общем случае не работает нумерация точек. Он может работать только при наличии гипотезы о том, что если есть разделение по кластерам, то она происходит упорядоченно с номером точки, например, при изучении кризиса теплообмена(на поверхности тепловыделяющего элемента образуется паровая пленочка, это приводит к резкому температурному скачку твэлла, т.е. расплавлению и выходу продуктов деления в теплоноситель). Делался эксперимент:




Увеличивая мощность, на каждой мощности получали аккустический спектор, т.е. в зависимости от частоты и амплитуды, мы видим, что где-то наступил кризис, это мы знаем из контроля температуры стенки твэлла.

Здесь метод цепных расстояний годится, если кластеры нормального теплообмена и кризис теплообмена существуют, то они должны естественно по мощности быть разделены, т.е. не может быть, чтоб первый спектор, который при маленькой мощности мерился - это кризис, а последний, котрый при большой - это норма. НЕТ! Может быть только норма, норма, норма, потом при увеличении мощности кризис, кризис, кризис и нормы уже быть не может.

!ВОПРОС! Почему мы берем какую-то книжку по Data Science, обязательно будет раздел кластерного анализа. Мы удивимся, что будет очень много методов. И будет очень много книг с подобным разделом и везде чертова туча алгоритмов кластерного анализа. ПОЧЕМУ ИХ ТАК МНОГО???

ЕСЛИ ЖЕНЩИН МНОГО, ТО ЗНАЧИТ НЕТ ОДНОЙ. Следовательно, если алгоритмов много, то значит нет одного, который решал бы с любыми данными задачу кластерного анализа. Алгоритмов много, потому что они зависят от ситуации и характеристик данных, которые мы имеем.

**Big Data - Большие??? данные.**

То что сегодня считается большим, завтра станет средним, после завтра - маленьким. Получился этот термин не по объему данных или скорости, а потому что данные с большого числа источников поступали непрерывно и их нужно было обрабатывать в темпе поступления, не сохраняя на диск (кодировка торговых данных, валют и т.п.)

На каких-то искусственных примерчиках, мы можем ручками что-то набрать. Но когда большой объем данных, ничего не возможно без компьютера, а конктретно - программы.

3. ***Программа.***

   Программу будем писать на самых популярных языках: Python и R.

   Они хороши своей относительной простотой в том плане, что здесь не нужно быть гиков, чтобы их использовать для анализа данных. Самое главное, что в них есть бесконечное множество библиотек, а библиотека - это целое множество программ для анализа данных. Это языки с открытым исходным кодом.

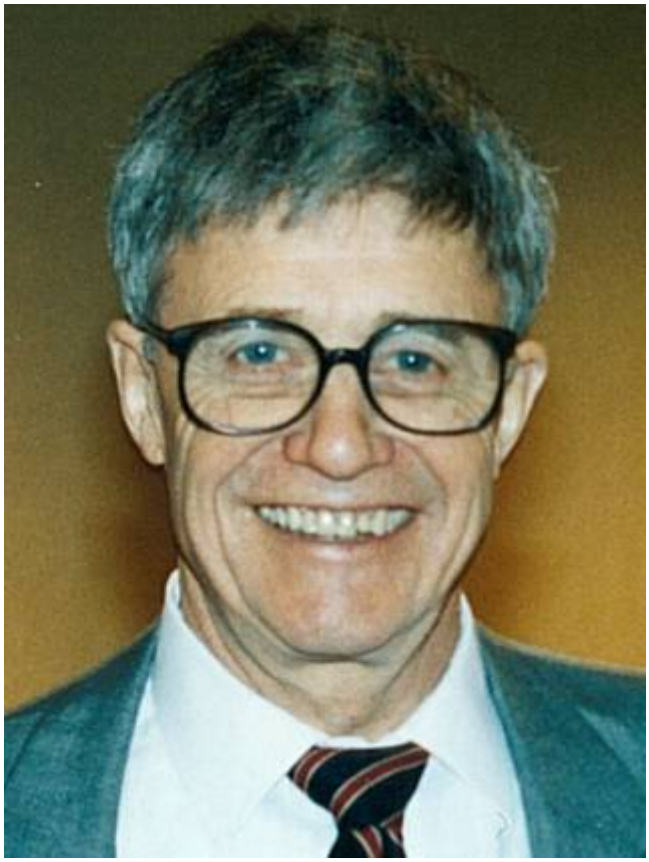
   Никто бы не использовал Python без расширения NumPy, который содержит мощь и красоту работы с массивами данных.

   R в отличии от Python - язык не общего назначения, на нем только анализируют данные.

   На вопрос, что лучше ставить? Ответ: BOTH!!! (Оба)

# APL

A(неопределенный артикль) Programming Language - какой-то язык программирования. Создал его велики программист и матиматик Кеннет Айверсон.



В окружающем мире существует много нотаций - какие-то принятые соглашения по поводу того, что тот или иной знак означает. Если бы все трактовалось буквами, было бы, например, много автомобильных аварий или электросхемы нужно было бы расписывать на кучи ватманоских листах вместо одного.

Или в арифметике, у нас есть пример 1+1=2, если бы не было нотации, мы бы писали: один плюс один равно два. УЖАС УЖАСНЫЙ!

Кеннет Айверсон. Он преподавал в университете преобразование информации для анализа данных. Устав стучать мелом по доске и долго-долго объяснять какие-то простые алгоритмы преобразования данных, психонул, сел и придумал краткую, выразительную математическую нотацию, которая позволяет любые преобразования данных компактно и наглядно записывать.

Он описал эту нотацию в книге, которая вышла в 1962 году под названием "A Programming Landuage". Самое смешное то, что Кеннет Айверсон, создавая эту нотацию и выпуская эту книгу,  не думал о компьютерах, не думал, что APL будет языком программирования, т.к. использовал программирование в другом смысле как передча или язык для записи алгоритмов. Эта книга имела взрывной успех по всему миру. Сразу появились книги: "Математическая статистика на APL", "Линейная алгебра на APL", где уже какие-то специализированные области математики и чего-то еще записывали именно на APL и показывали как это удобно, хорошо, коротко получается.

В 1966 году IBM пригласила Айверсона и еще несколько человек разработать на базе APL язык программирования, который на машинах IBM работал бы.

Это настолько знаменательное событие, что Кеннет Айверсон получил премию Тьюринга в 1979 году, которая приравнивается к премии Нобиля. А в 1982 году - премию компьютерного пионера как первопроходец. Они считаются как две самые великие премии, которые может получить программист или математик.

Но пока нас это не убеждает в том, что APL нам нужен для анализа данных.