

Database scaling and big data processing. Describe different approaches to database scaling (Vertical and Horizontal scaling, application level, mirroring, partitioning, sharding) and non-database approaches to big data storing and processing (DW, DataLakes, ETL/ELT).

Partitioning: Divide large database into smaller manageable pieces. each stored on different servers.

→ To improve performance Scalability of large databases

Database Scaling



Vertical Scaling



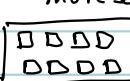
Scale up.
more memory
internal
resources

Horizontal Scaling

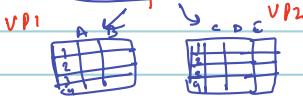
more servers

Application level

Mirroring



Partitioning



Vertical Horizontal

Vertical Scaling: Single server → ↑ Resources

- RAM
- CPU
- Storage (HDD)

to handle ↑ load.

- Typically used with relational DB MySQL, PostgreSQL.

Horizontal Scaling: Multiple servers → load distributed across several servers & each server works on part of the workload

- ↑s capacity by adding ↑ nodes
- Most common in NoSQL databases (Cassandra)
- SQL DB → possible with sharding
- In SQL, less common due to consistency requirements.

Application level scaling: Implemented at software/application level → to help manage ↑ load

Idea: to reduce burden on DB, and do handling of some tasks at the application layer

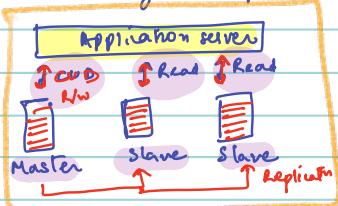
on both SQL and NoSQL

e.g.: Redis is used for cache → NH, NSP

Mirroring: keeping copies of database on multiple servers to ensure high availability.

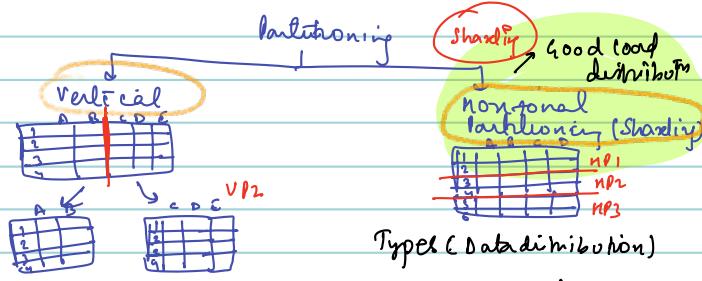
SQL → MySQL → for fault tolerance

NoSQL → MongoDB replica sets.



App: fast response for R Queries

Slow
slave queries
limited db



Columns of db = into partitions

Partition: ÷ within same database

Sharding: ÷ among multiple databases or across servers

Type of Data distribution

Range based: 1-100
100-200

Random

Data inner logic

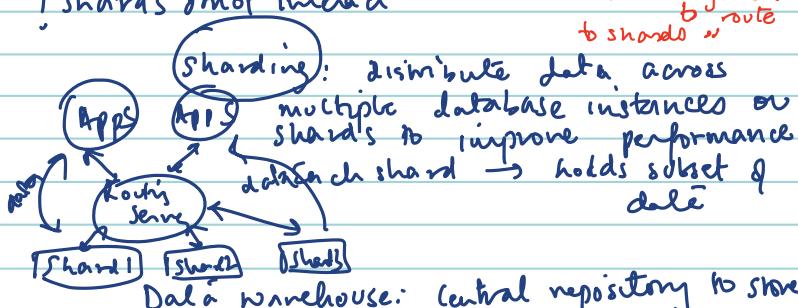
Geo spatial location

Math function

Relationship b/w shards
servers don't exist

or Management
to shard → route

shards don't interact



Data warehouse: central repository to store

large amt of structured data from multiple sources. Optimized for query and analysis & transaction processing



employs: star or snowflake schema

User of AP for complex queries

Data lake: large repo. to store unstructured, semi-structured, structured data in native form

Scheme on read: Data interpreted

eg. AWS S3, Azure Data Lake

ETL

- for structured
- small data set

slower

ELT

- all data types
- large data sets

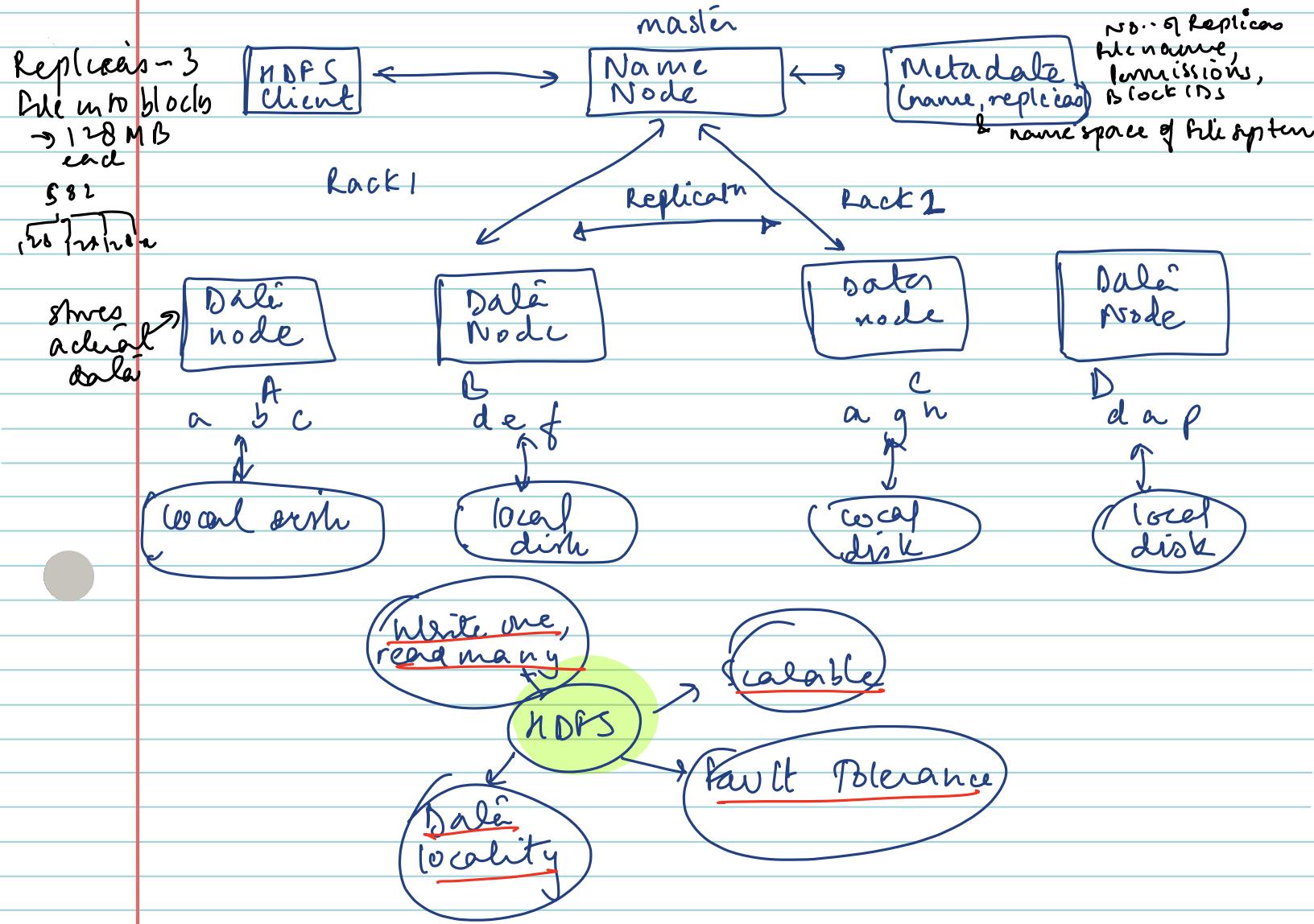
faster (using 1 today economy of data)

At storage layer

Distributed file storage

HDFS Architecture: Hadoop distributed file system

In big data (huge data), it is designed to handle large volume of data across distributed N/w of computers.

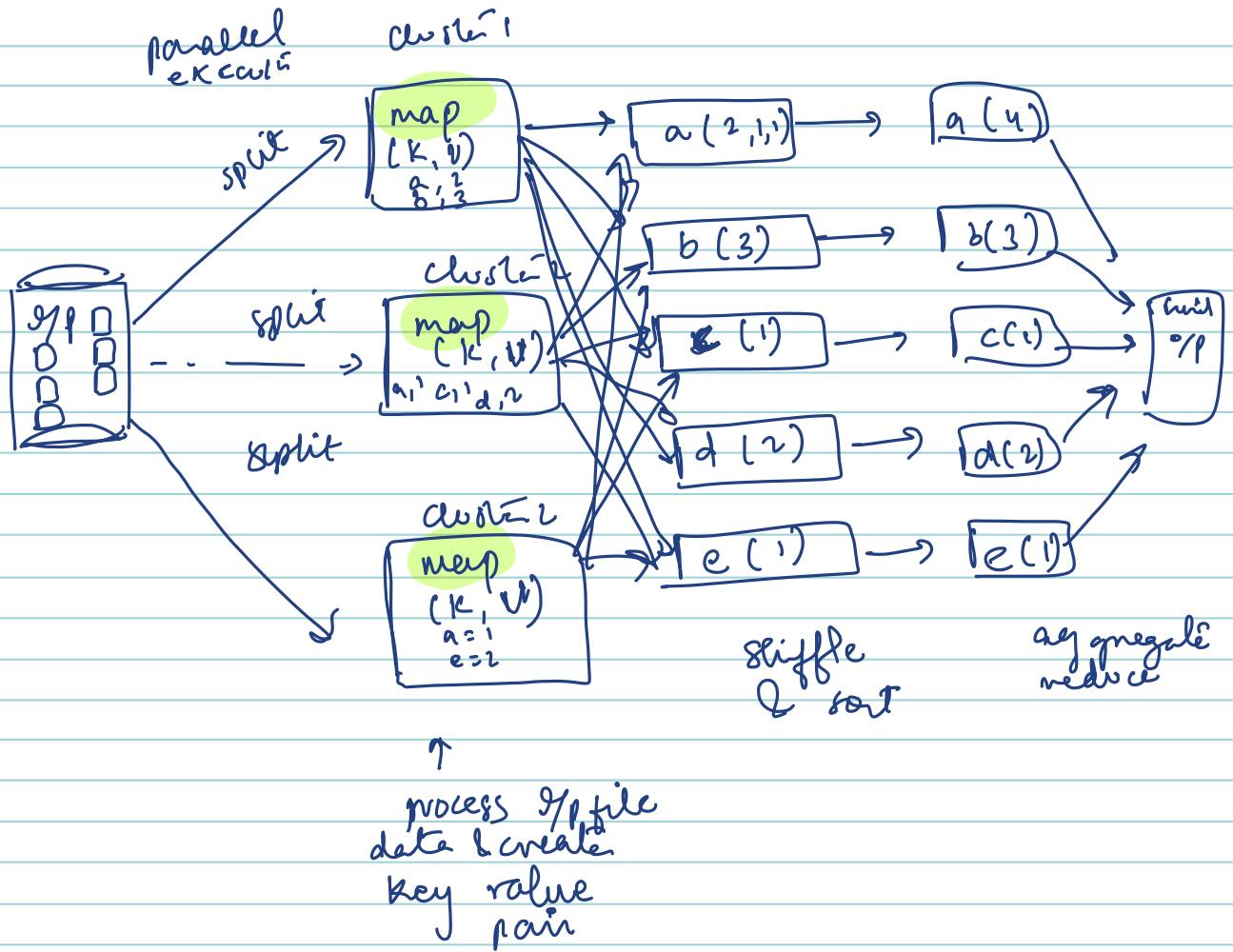


Application = Hadoop Map Reduce → for processing data layer:

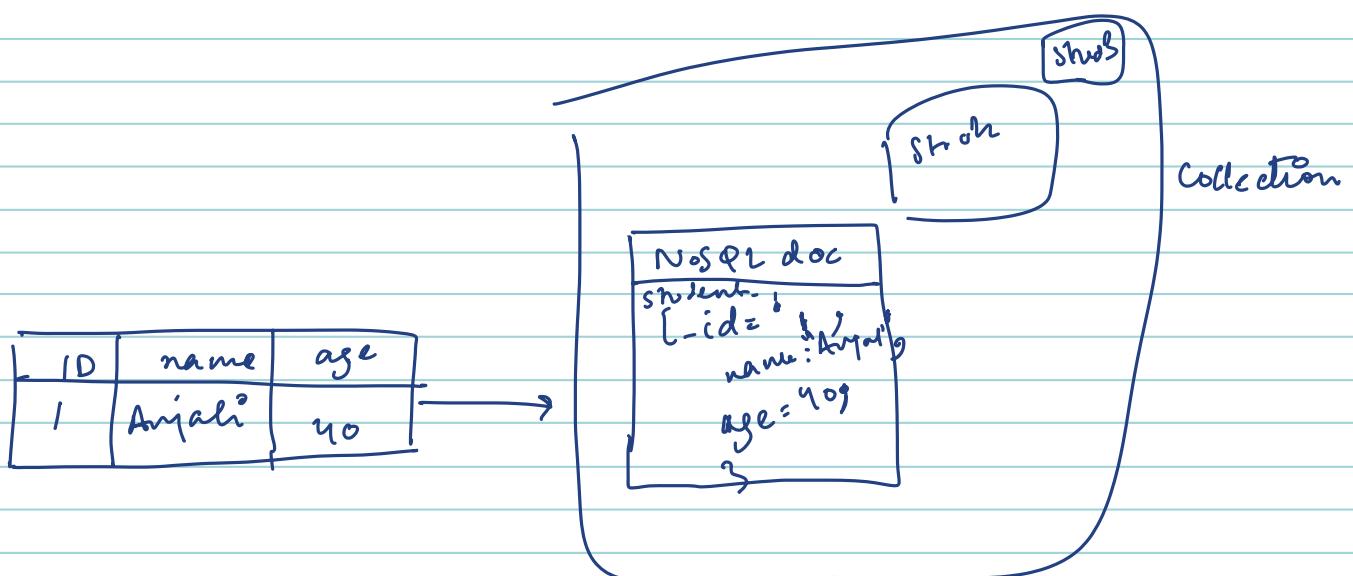
Map Reduce → Data processing technique for large datasets in a parallel manner.

- Used in Hadoop widely.
- Simplifies data processing across clusters of machines by dividing tasks into sub-tasks and executing them in parallel.

Count the occurrences of words in a document



C

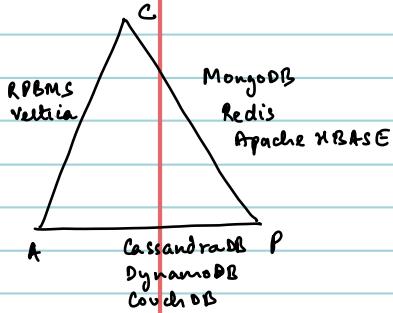


AP Theorem

[Brewer's theorem]

C = Consistency
 A = Availability
 P = Partition Tolerance.

All nodes see same data all the time
 Each request receives a response (eventually not)
 system continues to operate despite n/w partitions or conn failures b/w nodes



During Partition Tolerance [Network failure], either Consistency or Availability can be provided by a system (DB).

ACID

Atomicity
 consistency
 Isolation
 Durability

} property to ensure reliable transaction processing

Txn in

Atomicity: "All or Nothing"

If any part of txn fails the entire transaction is rolled back.
 Account A → Account B & no changes are applied.
 → sent 100 . 80 (Not updated) → Rolled back

Consistency: follows rules

brings db from one valid state to another valid state

eg: -ve balance (Rule Violation)
 Sum of total remains same

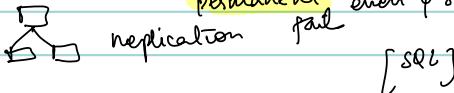
Isolation: Concurrent Txns are isolated from each other.

T1 | T2 | T3
 (RCA) (RCB) (RCA).

Customer withdrawing money from same account, isolation

Ensures that both the txns don't see each other
 ↳ It is always consistent

Durability: All changes in DB should be permanent even if system replication fails



[SQL]

BASE Properties

NoSQL

- B → Basically Available
- S → Soft State
- E → Eventually consistent

Basically Available: System guarantees availability.

DB will respond to all requests even if data sent is stale or incomplete

e.g:

Inventory system → sold out not updated on website during high traffic

Soft state

Even w/o input from outside, the state of system changes. Due to the eventual updates across distributed system

Ex:

After 1 minute

Eventually consistent: System will eventually become consistent given enough time.

For updates, it will make true (replicas)
 But eventually becomes consistent.

Types of NoSQL databases

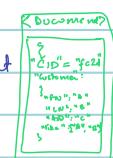
Key value

Based on key-value principle.
 Data accessed only using key (Index)
 least complex
 schemaless (plastic) (can remap easily)
 Example: Redis, Dynamo DB, Amazon



Document Database

Extension of Key Value
 Value is stored in document format (XML or JSON)
 more complex than KV pair
 key could be number, character
 Document with same structure are grouped into collections.



Example: MongoDB, Couch DB

"Best persisting catalogues"

Column Database

Work on column
 In Row DB (SQL) = 1, A, 22; 2, B, 33;
 3, C, 44

Data stored in column DB = 1, 2, 3; A, B, C
 Faster lookups

Avg min, max
 Analytics
 Speed

cassandra: example

Graph Database: for highly connected data eg: Social Media Graph DB used

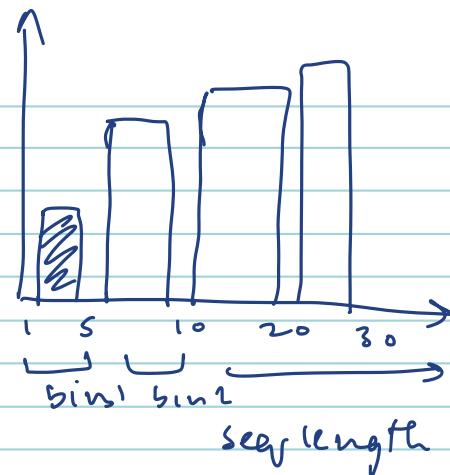


Example:
 Infinitely
 Graph

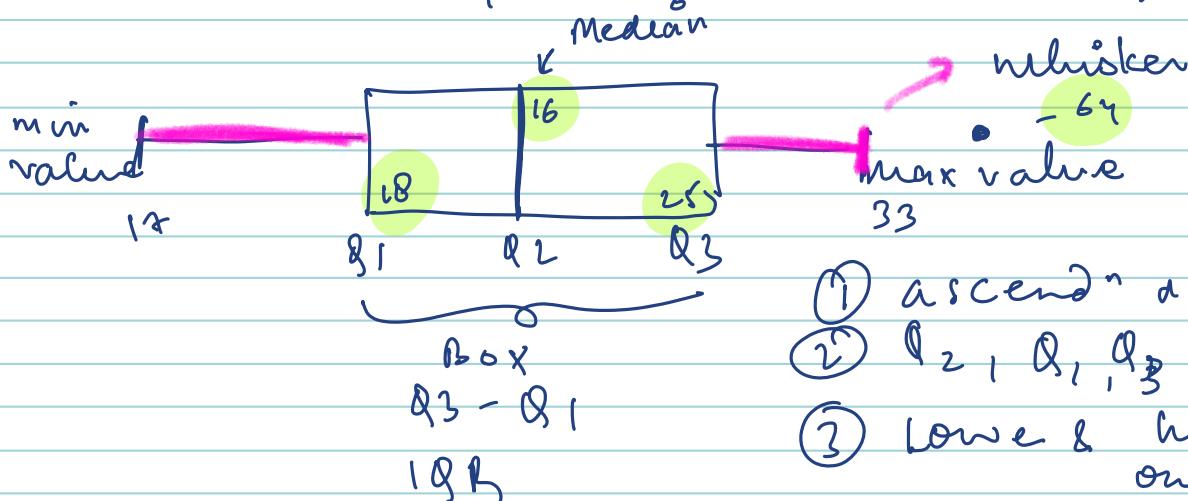
histogram:

- distribution of data by freq
÷ them into bins

- bin - show frequency of data points

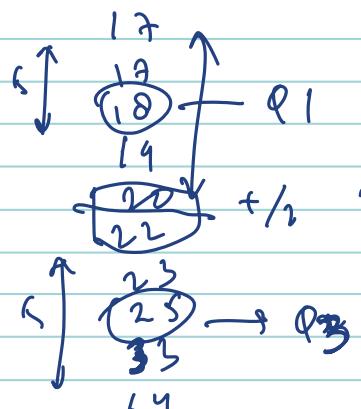


Box plot: Show dist of data (how data is spread)



- ① ascend data
- ② Q_1, Q_2, Q_3
- ③ lower & higher outlier

data in asc format



Q_1, Q_2, Q_3
Median

check
outlier [higher outlier]

$$= Q_3 + 1.5 \times IQR$$

$$= 25 + 10 \cdot 5$$

$$(25 - 18)$$

$$= Q_3 + 1.5 \times 10 \cdot 5$$

$$= 35.5$$

any data > 35.5 would
be outlier

$\therefore 64$ is outlier

② lower Outlier

$$= Q_1 - 1.5 \times IQR$$

$$= 18 - 10 \cdot 5 \Rightarrow 8.5$$

vi lower outlier

Covariance

$$\text{Cov}(X, Y) = \frac{1}{n-1} \sum_{i=1}^n (X_i - \bar{X}_i)(Y_i - \bar{Y}_i)$$

sample

Indicates direction of the relation ship b/w variables

$$\text{cov: } \begin{matrix} -\infty & 0 & \infty \\ \text{ht wt (kg)} & 143 \end{matrix}$$

$$\begin{matrix} \text{ht wt (g)} \\ \cdot 143153 \end{matrix}$$

can't tell about strength
only show +ve or -ve

$\rightarrow 1000 \times \text{larger}$ (not stronger though)

calculates
stronger
or
weaker

Correlation

$$\text{corr}(X, Y) = \frac{\text{Cov}(X, Y)}{\sigma(X)\sigma(Y)}$$

Indicates strength & direction of relationship b/w variables

$$\begin{matrix} \text{ht wt (kg)} \\ 0.922 \end{matrix}$$

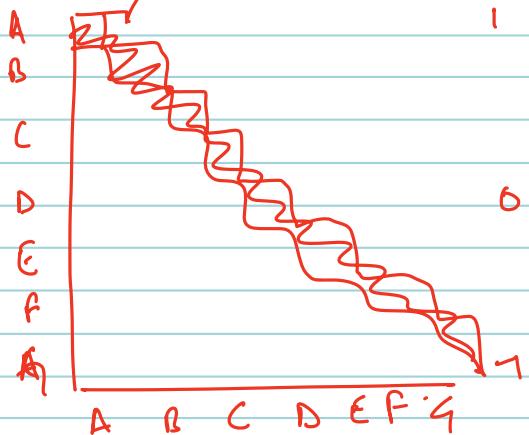
$$\begin{matrix} \text{ht wt (g)} \\ 0.922 \end{matrix}$$

\rightarrow same correlat'

$$-1 \leq \text{corr}(X, Y) \leq 1$$

If $r = 0$ = no relat

-1 \rightarrow strong negative
1 \rightarrow strong positive



does not
influence
by scale:
standardizes data