

Python for Data processing

Lecture 5: **Pandas - Part II**

Kosta Rozen

What we already know

- NumPy
- PyTorch
- basics of Pandas: series and dataframes, reading files

This lecture

- SQL-like operations on dataframes
(including grouping and joins)
- efficient Pandas

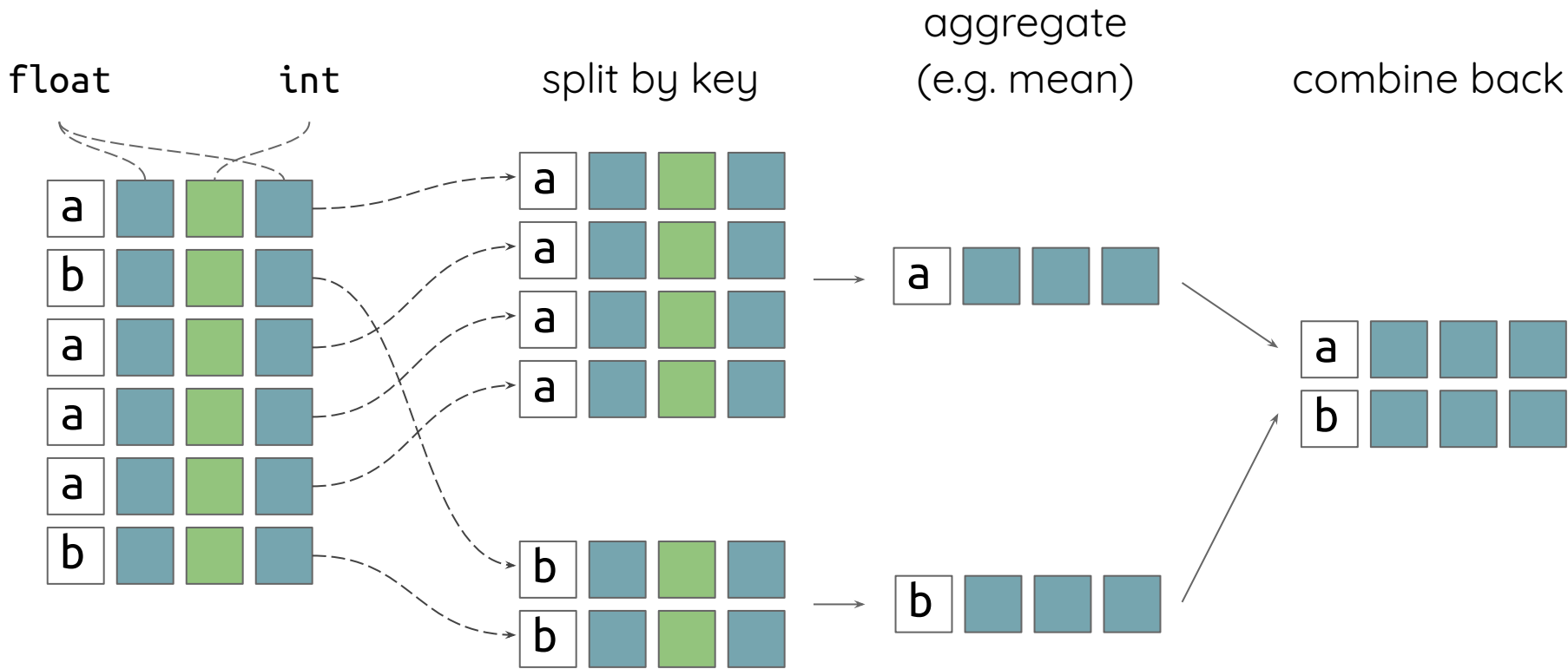
Grouping dataframes

Split-apply-combine

- group data by some key (keys)
- perform calculations for each group
- combine data

Very **common** pipeline

Split-apply-combine



Split-apply-combine

- not all operations may be applied to some columns
- columns **dtype** may change
- aggregation may not involve data values (`.size()`)

Basic grouping

Entry point to grouping operations in Pandas:

`df.groupby(...)`

Nothing happens immediately

Only when you **actually** perform operations

Grouper

Fancy way of setting grouping keys

Not very useful with usual keys

Extremely useful with **DateTime** keys

Joining dataframes

Join operations

Combine two dataframes based on some keys (in both)

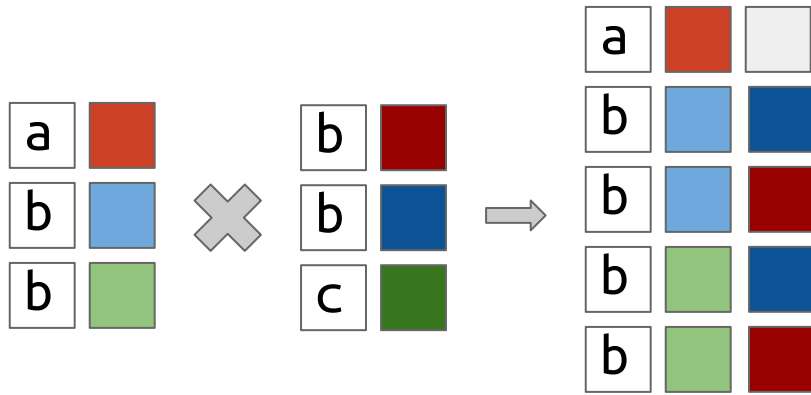
Types:

inner, outer, left, right

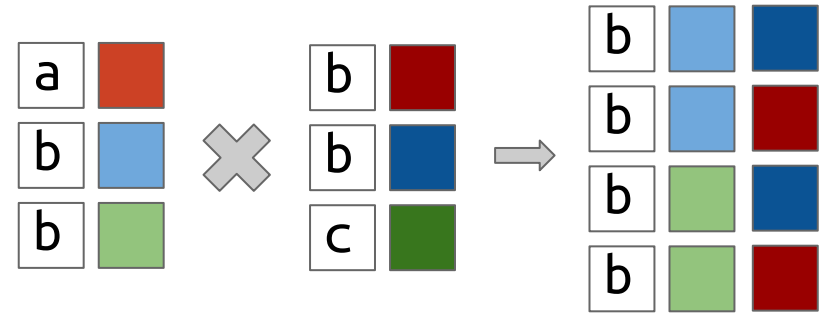
`df.join(other_df, on=...)` - simple join on indexes only

`df.merge(other_df, ...)` - more flexible join on any column

Left and inner joins

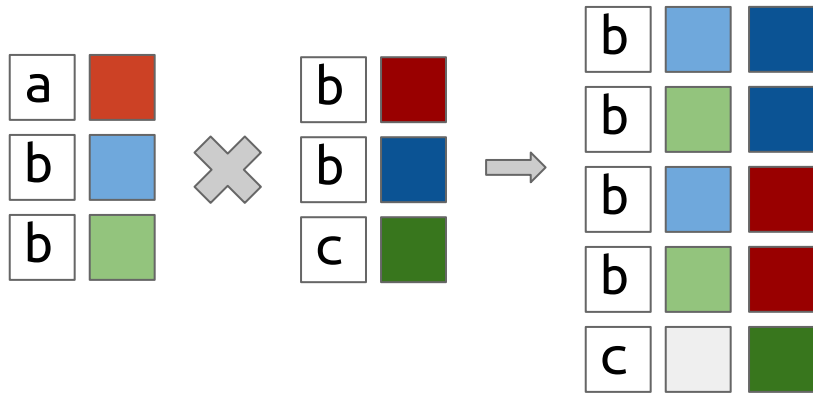


left join

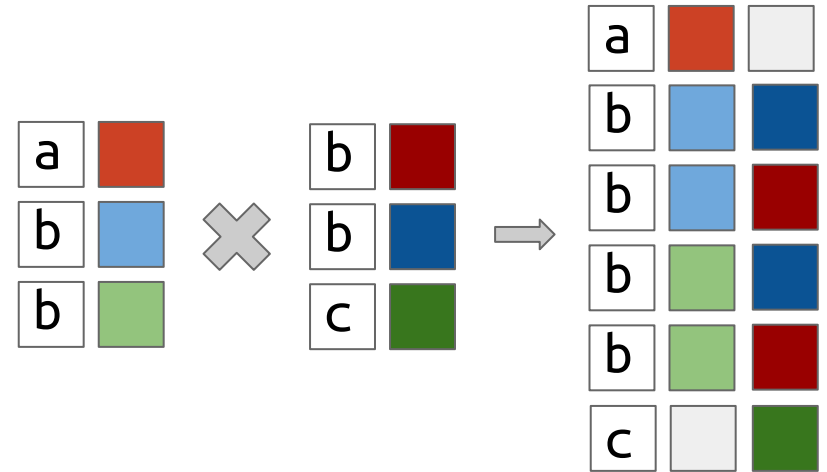


inner join

Right and outer joins



right join



outer join

Efficient pandas

Efficiency & performance

Two dimensions:

- running time
- memory footprint

Running time:

efficient loops, vectorized operations (it's often NumPy under the hood)

Memory:

reasonable dtype, no intermediate df's, **eval** and **query**

What we've learned

More advanced Pandas stuff:

- grouping
- joins
- efficient pandas

questions?