across

(Column-wise operations)

across?

- If we would like to **alter** or **aggregate multiple columns** at **once**, we can use dplyr's verbs:
  - **mutate_all() / summarise_all()** ~ ***_all** **all** columns affected
  - **mutate_at() / summarise_at()** ~ ***_at** **listed** columns affected
  - **mutate_if() / summarise_if()** ~ ***_if** columns that **meet condition**
- After some years, dplyr's developers have introduced a **new paradigm** packed in a verb called **across()**.
- The main idea of **across()** is to do **column-wise operations more generic** (simpler R syntax!).
- **across()** replaces all the old verbs (**summarise** and **mutate**) ending with suffix (***_all**, ***_at**, ***_if**).
- Old code should still work, but when writing new code we should consider **across()**!

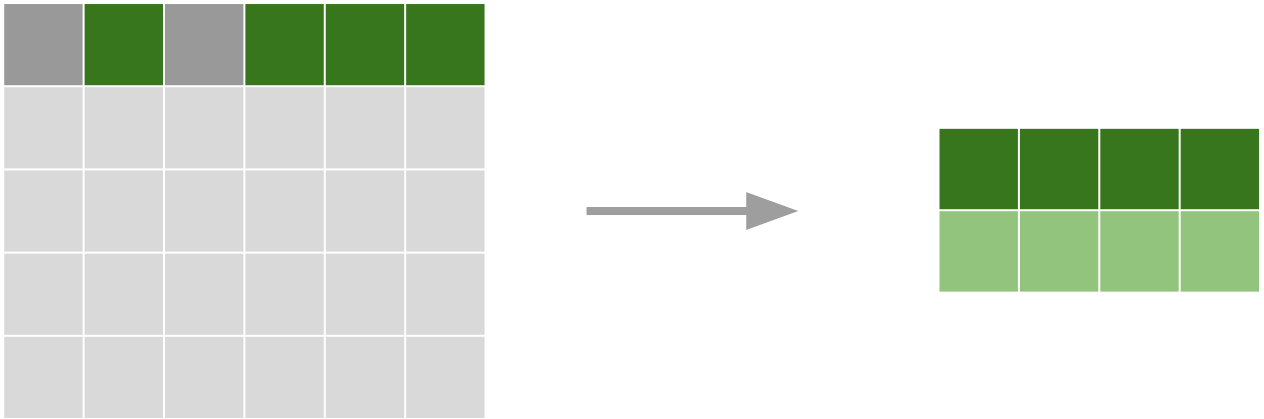
dplyr : : across & summarise

- Apply an aggregation function(s) across multiple columns and create a new table of summary statistics.

```
summarise( across(.cols,
                  .fns,
                  ...) )
```

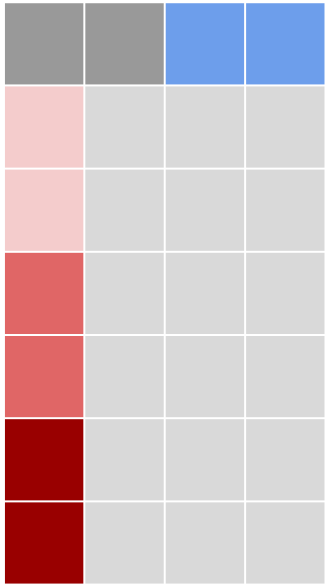
Columns to transform (tidy-select style)

Function(s) to be applied (function, purr-style function, or list of multiple functions)

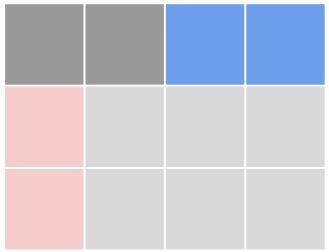


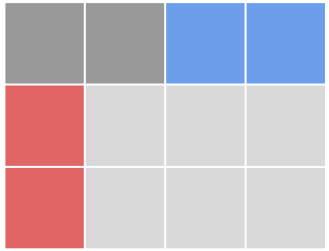
dplyr : : **across & summarise** with **group_by**

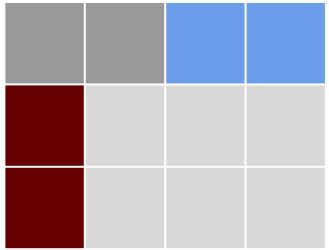
- Create “grouped” copy of a table.
- You group summarization operation per one or more “grouped” variable(s), only on selected columns (defined inside across verb)!



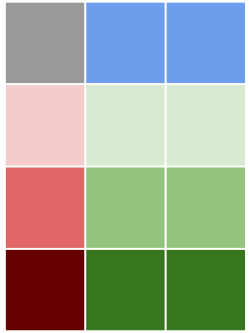
group_by()







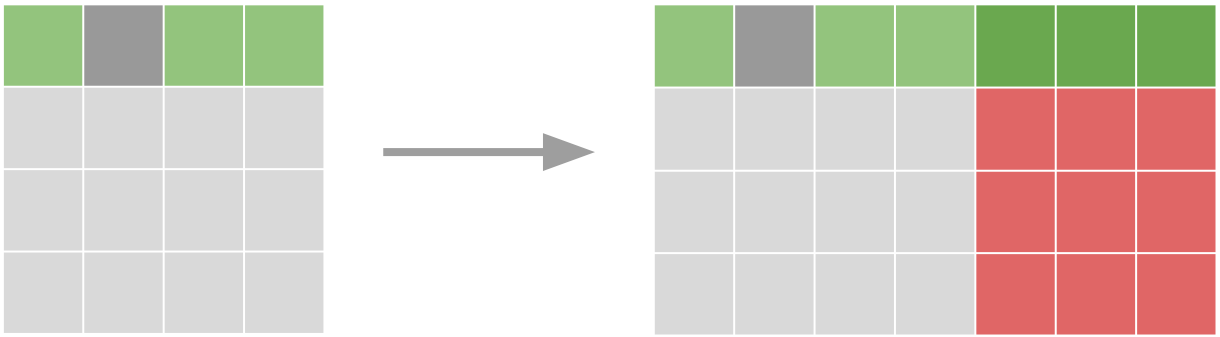
summarise(
across())



dplyr : : across & mutate

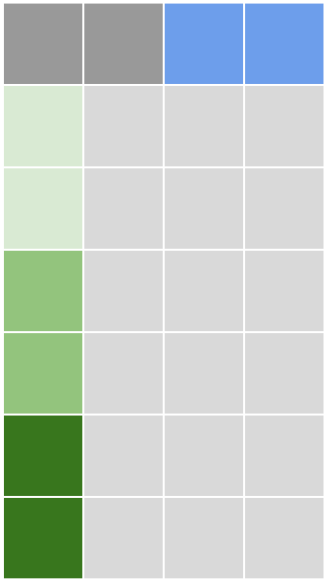
- Alter columns by applying selected function(s) across given columns.

```
mutate(across(.cols,
              .fns,
              ...) ))
```



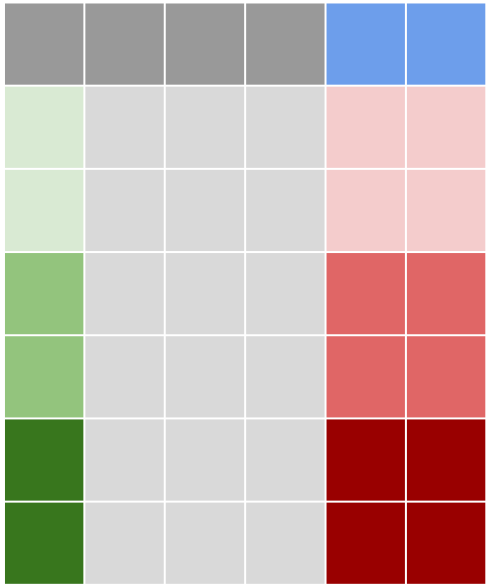
dplyr : : **across & mutate** with **group_by**

- Apply mutate operation (alter columns) using “grouped” variables across multiple columns!



group_by()

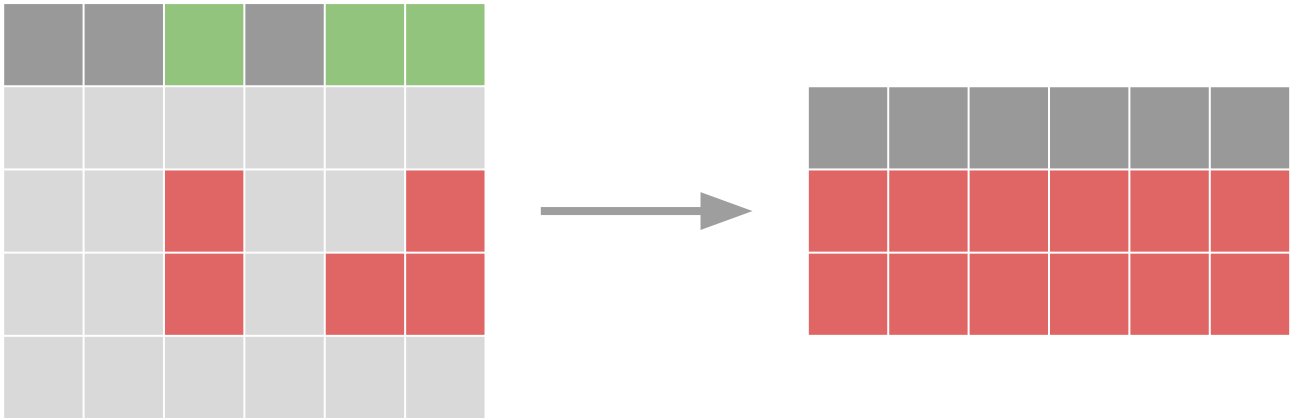
mutate(**across**())



dplyr : : if_any & filter

- Subset (filter) rows based on logical conditions considering multiple columns at once. **if_any()** keeps the rows where the **predicate** is **true** for **at least one selected column**!

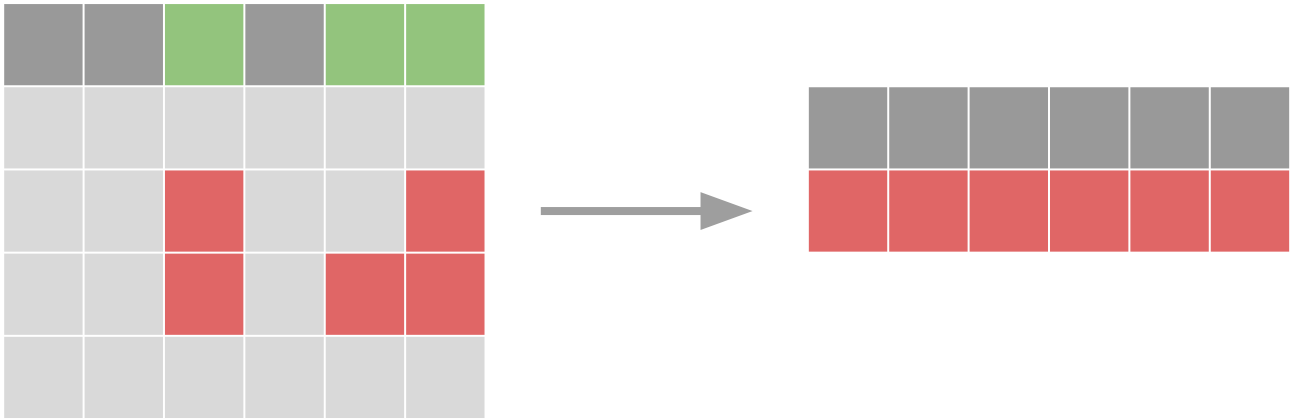
```
filter(if_any(.cols,
              .fns,
              ...))
```



dplyr : : if_all & filter

- Subset (filter) rows based on logical conditions considering multiple columns at once. **if_all()** keeps the rows where the **predicate** is **true** for **all selected columns**!

```
filter(if_all(.cols,
              .fns,
              ...))
```



Sources

- https://dplyr.tidyverse.org/reference/across.html
- https://dplyr.tidyverse.org/articles/colwise.html