# Data wrangling across columns

## Agenda and reminders

- HW 2 due Friday **on GitHub classroom**
  - Commit early and often, let me know if you have any technical problems
  - Make sure to submit both the `.qmd` **and** `md` files
- Department seminar tomorrow (9/11) at 11am in ZSR auditorium
  - please refrain from wearing colognes, perfumes, and/or heavily scented body and hair products
- Today: data wrangling across columns

## Warmup activity

Work on the activity (handout) with a neighbor, then we will discuss as a class

```
diamonds |>
    summarize( mean_carat = mean(carat),
               sd_ carat = sd (carat),
               mean_depth = mean(depth),
               ....
             )
```

## Warmup

```r
diamonds |>
  summarize(mean_carat = mean(carat),
            sd_carat = sd(carat),
            mean_depth = mean(depth),
            sd_depth = sd(depth),
            mean_price = mean(price),
            sd_price = sd(price))
```

```
# A tibble: 1 × 6
  mean_carat sd_carat mean_depth sd_depth mean_price sd_price
       <dbl>    <dbl>      <dbl>    <dbl>      <dbl>    <dbl>
1      0.798    0.474       61.7     1.43      3933.    3989.
```

Are there any downsides to this code?

## Warmup

```r
diamonds |>
  summarize(mean_carat = mean(carat),
            sd_carat = sd(carat),
            mean_depth = mean(depth),
            sd_depth = sd(depth),
            mean_price = mean(price),
            sd_price = sd(price))
```

- more variables to summarize means longer code, harder to read
- requires a lot of copying and pasting
- more prone to errors when typing names of functions, variables, etc.

## `across`: Data wrangling across columns

Instead of copying the same function multiple times for different columns, we can apply functions *across* the columns of a table:

columns to summarize

```r
diamonds |>
  summarize(across(c(carat, depth, price),
                   mean))
```

function to apply to each column

```
# A tibble: 1 × 3
  carat depth price
  <dbl> <dbl> <dbl>
1 0.798  61.7 3933.
```

## `across`: Data wrangling across columns

Instead of copying the same function multiple times for different columns, we can apply functions *across* the columns of a table:

```r
diamonds |>
  summarize(across(c(carat, depth, price),
                   mean))
```

```
# A tibble: 1 × 3
  carat depth price
  <dbl> <dbl> <dbl>
1 0.798  61.7 3933.
```

What if I want to calculate both the mean *and* the standard deviation of these columns?

## `across` with multiple functions

```r
diamonds |>
  summarize(across(c(carat, depth, price),
                   list(mean, sd)))
```

list of all the functions to apply

```
# A tibble: 1 × 6
  carat_1 carat_2 depth_1 depth_2 price_1 price_2
    <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
1   0.798   0.474    61.7    1.43   3933.   3989.
```

all combinations of columns & functions

What if I want to include the function name in the summary columns?

## `across` with multiple functions

```r
diamonds |>
  summarize(across(c(carat, depth, price),
                   list("mean" = mean, "sd" = sd)))
```

name          functions

```
# A tibble: 1 × 6
  carat_mean carat_sd depth_mean depth_sd price_mean price_sd
       <dbl>    <dbl>      <dbl>    <dbl>      <dbl>    <dbl>
1      0.798    0.474       61.7     1.43      3933.    3989.
```

What if I want to change the order of the column names (e.g. `mean_carat` vs. `carat_mean`)?

## `across` with multiple functions

```
diamonds |>
  summarize(across(c(carat, depth, price),
                   list("mean" = mean, "sd" = sd),
                   .names = "{.col}_{.fn}"))
```

← how do we combine column names + function names?

extract column name

function name

```
# A tibble: 1 × 6
  carat_mean carat_sd depth_mean depth_sd price_mean price_sd
       <dbl>    <dbl>      <dbl>    <dbl>      <dbl>    <dbl>
1      0.798    0.474       61.7     1.43      3933.    3989.
```

```
diamonds |>
  summarize(across(c(carat, depth, price),
                   list("mean" = mean, "sd" = sd),
                   .names = "{.fn}_{.col}"))
```

```
# A tibble: 1 × 6
  mean_carat sd_carat mean_depth sd_depth mean_price sd_price
       <dbl>    <dbl>      <dbl>    <dbl>      <dbl>    <dbl>
1      0.798    0.474       61.7     1.43      3933.    3989.
```

## Summarizing more columns

```r
diamonds |>
  summarize(across(c(carat, depth, price),
                   list("mean" = mean)))
```

```
# A tibble: 1 × 3
  carat_mean depth_mean price_mean
       <dbl>      <dbl>      <dbl>
1      0.798       61.7      3933.
```

How would I modify this code to calculate the mean for *all* the numeric variables (`carat`, `depth`, `table`, `price`, `x`, `y`, `z`)?

## Summarizing more columns

Option 1:

```r
diamonds |>
  summarize(across(c(carat, depth, table, price, x, y, z),
                   list("mean" = mean)))
```

```
# A tibble: 1 × 7
  carat_mean depth_mean table_mean price_mean x_mean y_mean z_mean
       <dbl>      <dbl>      <dbl>      <dbl>  <dbl>  <dbl>  <dbl>
1      0.798       61.7       57.5      3933.   5.73   5.73   3.54
```

Are there any issues with this approach?

- prone to errors
- tedious
- time consuming

⋮

## *Efficiently* summarizing more columns

find columns that satisfy a condition

checking if column is a numeric variable

```r
diamonds |>
  summarize(across(where(is.numeric),
                   list("mean" = mean)))
```

```
# A tibble: 1 × 7
  carat_mean depth_mean table_mean price_mean x_mean y_mean z_mean
       <dbl>      <dbl>      <dbl>      <dbl>  <dbl>  <dbl>  <dbl>
1      0.798       61.7       57.5      3933.   5.73   5.73   3.54
```

## *Efficiently* summarizing more columns

```r
diamonds |>
  summarize(across(where(is.numeric),
                   list("mean" = mean)))
```

```
# A tibble: 1 × 7
  carat_mean depth_mean table_mean price_mean x_mean y_mean z_mean
       <dbl>      <dbl>      <dbl>      <dbl>  <dbl>  <dbl>  <dbl>
1      0.798       61.7       57.5      3933.   5.73   5.73   3.54
```

`where(is.numeric)` returns the columns which are numeric:

```r
is.numeric(diamonds$carat)
```

```
[1] TRUE
```

```r
is.numeric(diamonds$price)
```

```
[1] TRUE
```

```r
is.numeric(diamonds$clarity)
```

```
[1] FALSE
```

## *Efficiently* summarizing more columns

We can use `where` with other functions too. For example:

find the factor variables

```
diamonds |>
  summarize(across(where(is.factor),
                   list("num_categories" = n_distinct)))
```

number of distinct values

```
# A tibble: 1 × 3
  cut_num_categories color_num_categories clarity_num_categories
               <int>                <int>                  <int>
1                  5                    7                      8
```

What do you think this code is doing?

## Class activity

https://sta279-f25.github.io/class_activities/ca_07.html

- Work with a neighbor on the class activity
- At the end of class, submit your work as an HTML file on Canvas (one per group, list all your names)

**For next time**, read:

- Chapter 25.2 in *R for Data Science*