Intro to Iteration

Class activity

https://sta279-f25.github.io/class_activities/ca_11.html

- Work with a neighbor on the class activity
- We will spend the first portion of today on the activity, then we will discuss as a class
- At the end of class, submit your work as an HTML file on Canvas (one per group, list all your names)

Iteration motivation

What are some potential issues with the following code?

```r
read_csv("intro_stats_grades/section_1.csv") |>
  slr_slope(midterm_1, midterm_2)

read_csv("intro_stats_grades/section_2.csv") |>
  slr_slope(midterm_1, midterm_2)

read_csv("intro_stats_grades/section_3.csv") |>
  slr_slope(midterm_1, midterm_2)
```

purrr::map

```r
grade_files <- list.files("intro_stats_grades", full.names=T)
grade_tables <- map(grade_files, read_csv)
```

What is the `map` function doing here?

`purrr::map`

```r
grade_tables <- map(grade_files, read_csv)
```

`map`: apply a function to each element of a list or vector

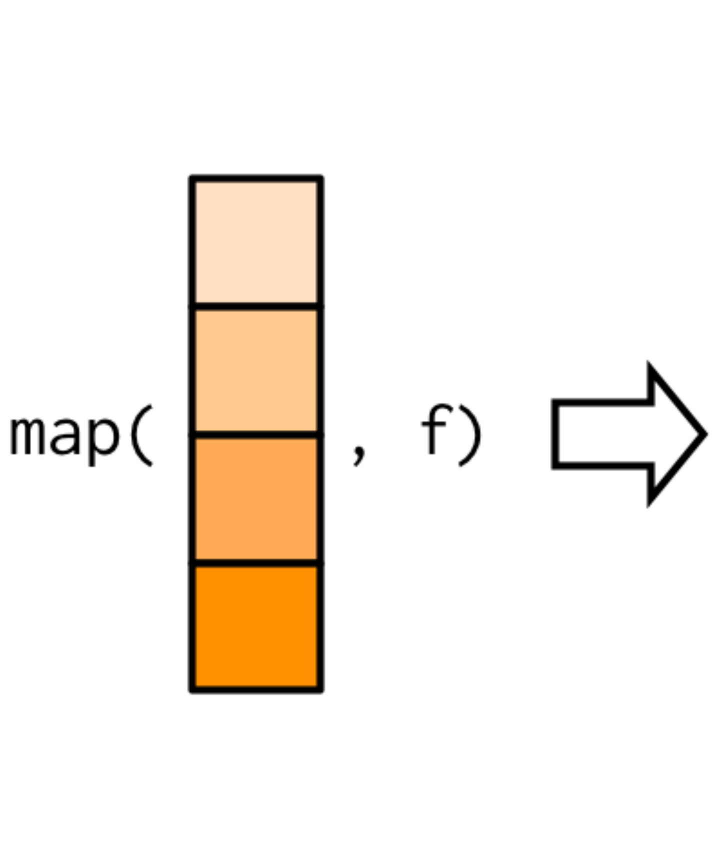
- first argument: a list or vector
  - `grade_files`: a vector of CSV file names to read into R
- second argument: the function to apply
  - `read_csv`: function to read a CSV file into R

"For each file in `grade_files`, apply the `read_csv` function to read it into R"

purrr::map

input: file name (string);
path to a csv file
on your machine

```r
grade_tables <- map(grade_files, read_csv)
```



(Image from *Advanced R* (2nd edition), Chapter 9)

purrr::map

```r
grade_files <- list.files("intro_stats_grades", full.names=T)
grade_tables <- map(grade_files, read_csv)
```

`map`: apply a function to each element of a list or vector

Output: a list

```r
typeof(grade_tables)
```

```
[1] "list"
```

```r
length(grade_tables)
```

```
[1] 10
```

```r
glimpse(grade_tables[[1]])
```

```
Rows: 35
Columns: 14
$ student_id <dbl> 55817, 32099, 40295, 54195, 15297, 81786, 49747,
78226, 102…
$ hw_1       <dbl> 10, 10, 10, 10, 10, 7, 10, 10, 9, 9, 8, 10, 10, 7,
8, 8, 10…
```

```
$ hw_2       <dbl> 10, 9, 10, 9, 8, 8, 9, 9, 9, 8, 10, 10, 10, 6, 9,
10, 8, 10…
$ hw_3       <dbl> 9, 10, 9, 9, 9, 6, 8, 9, 10, 10, 8, 9, 9, 9, 10, 9,
10, 8, …
$ hw_4       <dbl> 9, 9, 9, 6, 10, 6, 8, 10, 7, 9, 9, 10, 10, 9, 9, 8,
9, 10, …
```

purrr::map

```r
grade_files <- list.files("intro_stats_grades", full.names=T)
grade_tables <- map(grade_files, read_csv)
```

`map`: apply a function to each element of a list or vector

Output: a list

```r
glimpse(grade_tables[[2]])
```

```
Rows: 29
Columns: 10
$ student_id <dbl> 88275, 99752, 81485, 34888, 56497, 14363, 31087,
34334, 278…
$ hw_1       <dbl> 8, 8, 10, 4, 5, 7, 5, 10, 10, 7, 6, 7, 7, 9, 9, NA,
NA, 7, …
$ hw_2       <dbl> 6, 10, 9, 5, 8, 7, 8, 9, 10, NA, 8, 10, 8, NA, 10,
10, 8, 6…
$ hw_3       <dbl> 8, 10, 9, 6, 7, 10, 6, 7, 10, 10, 5, 10, 8, 7, 9, 8,
7, 8, …
$ hw_4       <dbl> 10, 10, 9, 9, 7, 9, 4, 8, 10, 7, 7, 8, 9, 9, 9, 9,
NA, 7, 1…
```

Another example

```r
x <- c(1, 4, 9, 16, 25)
map(x, sqrt)
```

What will this code produce?

Another example

```r
x <- c(1, 4, 9, 16, 25)
map(x, sqrt)
```

```
[[1]]
[1] 1

[[2]]
[1] 2

[[3]]
[1] 3

[[4]]
[1] 4

[[5]]
```

map variants

If we want to return a vector instead of a list, we can use one of the map variants. E.g.:

```r
x <- c(1, 4, 9, 16, 25)
map_dbl(x, sqrt)
```

```
[1] 1 2 3 4 5
```

make output a numeric vector

Another approach:

sqrt(x) (vectorized operation)

↳ 1 2 3 4 5

Another example

```r
map_dbl(1:10, function(x) x + 1)
```

What will this code produce?

Another example

```r
map_dbl(1:10, function(x) x + 1)
```

```
 [1]  2  3  4  5  6  7  8  9 10 11
```

Class activity

```r
slr_slope <- function(df, x, y) {
  df |>
    summarize(slope = cov({{ x }}, {{ y }}, use="complete.obs")/
                      var({{ x }}, na.rm=T))
}

list.files("intro_stats_grades", full.names=T) |>
  map(read_csv) |>
  map(slr_slope)    ← doesn't specify x, y variables
```

```
Error in `map()`:
ℹ In index: 1.
Caused by error in `summarize()`:
ℹ In argument: `slope = cov(, , use = "complete.obs")/var(, na.rm =
T)`.
Caused by error in `cov()`:
! is.numeric(x) || is.logical(x) is not TRUE
```

What is causing this error?

Class activity

```r
slr_slope <- function(df, x, y) {
  df |>
    summarize(slope = cov({{ x }}, {{ y }}, use="complete.obs")/
                      var({{ x }}, na.rm=T))
}

list.files("intro_stats_grades", full.names=T) |>
  map(read_csv) |>
  map(function(df) slr_slope(df, midterm_1, midterm_2))
```

only take in one argument

always compute slope for midterm_1 & midterm_2

```
[[1]]
# A tibble: 1 × 1
  slope
  <dbl>
1 0.756

[[2]]
# A tibble: 1 × 1
  slope
  <dbl>
1 0.871
```

[[3]]

`purrr::map`

The function to be applied in `map` must take a single argument

```r
# slr_slope takes THREE arguments:
list.files("intro_stats_grades", full.names=T) |>
  map(read_csv) |>
  map(slr_slope)
```

```r
# the anonymous function takes only ONE argument:
list.files("intro_stats_grades", full.names=T) |>
  map(read_csv) |>
  map(function(df) slr_slope(df, midterm_1, midterm_2))
```

Another example

```r
ex_list <- list(
  c(1, 2, 3),
  c(2, 3, 4)
)

map_dbl(ex_list, mean)
```

What do you think will be the output of this code?

Another example

```r
ex_list <- list(
  c(1, 2, 3),
  c(2, 3, 4)
)

map_dbl(ex_list, mean)
```

```
[1] 2 3
```

```r
ex_list[[1]]
```

```
[1] 1 2 3
```

```r
mean(ex_list[[1]])
```

```
[1] 2
```

```r
ex_list[[2]]
```

```
[1] 2 3 4
```

```r
mean(ex_list[[2]])
```

```
[1] 3
```

Another example

```r
ex_list <- list(
  c(1, 2, NA),
  c(2, 3, 4)
)

map_dbl(ex_list, mean)
```

What do you think will be the output of this code?

Another example

```r
ex_list <- list(
  c(1, 2, NA),
  c(2, 3, 4)
)

map_dbl(ex_list, mean)
```

```
[1] NA  3
```

How do we ignore the NA when calculating the mean?

Another example

```r
ex_list <- list(
  c(1, 2, NA),
  c(2, 3, 4)
)

map_dbl(ex_list, mean(na.rm=T))
```

Will this code work?

Another example

```r
ex_list <- list(
  c(1, 2, NA),
  c(2, 3, 4)
)
                 function(x)  mean(x, na.rm=T)
map_dbl(ex_list, ~~mean(na.rm=T)~~)
```

```
Error in mean.default(na.rm = T): argument "x" is missing, with no
default
```

Problem: `mean(na.rm=T)` is not a function! It is a *call* to the `mean` function.

Solution: use an anonymous function!

Another example

```r
ex_list <- list(
  c(1, 2, NA),
  c(2, 3, 4)
)

map_dbl(ex_list, function(x) mean(x, na.rm=T))
```

```
[1] 1.5 3.0
```